Semantic Routing in Peer-to-Peer Systems

Siebes, R.M.

2006

document version
Publisher's PDF, also known as Version of record

SIKS Dissertation Series No. 2006-10

The research reported in this thesis has been carried out under the auspices of SIKS, the Dutch Graduate School for Information and Knowledge Systems.

Promotiecommissie:
prof.dr. F. A. H. van Harmelen (promotor)
prof.dr. K. Aberer (Swiss Federal Institute of Technology Lausanne)
prof.dr. F. Giunchiglia (University of Trento)
prof.dr. A. T. H. Schreiber (Vrije Universiteit Amsterdam)
prof.dr. M. van Steen (Vrije Universiteit Amsterdam)

Cover illustration by Theo Siebes, Andries Ouwendorp en Ronny Siebes

VRIJE UNIVERSITEIT

Semantic Routing in Peer-to-Peer Systems

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad Doctor aan
de Vrije Universiteit Amsterdam,
op gezag van de rector magnificus
prof.dr. T. Sminia,
in het openbaar te verdedigen
ten overstaan van de promotiecommissie
van de faculteit der Exacte Wetenschappen
op vrijdag 9 juni 2006 om 10.45 uur
in de aula van de universiteit,
De Boelelaan 1105

door

Ronald Maria Siebes

geboren te Doetinchem

promotor: prof.dr. F.A.H. van Harmelen

Contents

preface

The movement of a wing from a butterfly can be the cause of a hurricane, an example of a chain of physical events combined with a large amount of unsafely stored energy. I think that a scientific process also works like this; a small conversation with a stranger can be the begin of a long chain of events, and after a lot of energy can lead to a thesis. People who we talk with and things that we see on TV or read in books and discuss about, even outside the science domain, paves and reshapes our research paths leading to eventual results. Our Western style of doing research is very focussed on individual achievements, which leads to the problem that we are attempted to optimize and focus only on our small part in the whole picture which is not necessarily good for global scientific progress and the wellbeing of everything that lives on our planet and beyond. Face recognition, data-mining, army-robots are examples which, when used by a malicious regime, can be threatening for people and other creatures. Also popular search engines use AI techniques learned at universities to make profiles of users and to sensor specific data (like the deal with an Asian country to censor content on human rights). Luckily people are starting to be more concerned about privacy and censorship issues and are working on systems that lowers this problem. The message is clear: be careful with technology. What could help the 'global' science and increase social responsibility is to lower the barrier to participate the scientific process, ie. make it a public event where everyone can contribute and criticize in an open setting. In this way people can be warned about potential threats in an early stage and allow other researchers start working on things that lowers the danger. To keep the process manageable, research on (semi) automatic reputation management provides means to distinct between high and low quality contributions. The initiator of thinking about lowering the boundaries of science participation is Chide Groenouwe and I am very grateful for his enthusiastic way of giving me these insights in collaborative science doing. I hope that the systems where my colleagues and I worked on will contribute to this process of scientific openness, by decentralizing content search process which prevents censorship and privacy restrictions. I'm very grateful to Frank van Harmelen: my promotor, money-arranger, colleague and teacher (random order of importance). My strongest love and gratefulness are to my dear parents and my brother Edwin, who always unconditionally supported me. Larissa, my beloved, I'm astonished and happy about your patience with me. Andries, Spyros, Sophie, nature, animals, friends, the reading committee and other colleagues, family, strangers that I met, thanks! Those who can read, enjoy reading.

*I*t’s not what you thought
When you first began it
You got what you want
Now you can hardly stand it though
By now you know its not
Going to stop
It’s not going to stop
It’s not going to stop ’til you wise up

You’re sure there’s a cure
And you have finally found it
You think one drink
Will shrink you ’til
You’re underground and living down
But it’s not going to stop
It’s not going to stop
It’s not going to stop ’til you wise up

Prepare a list of what you need
Before you sign away the deed
’couse it’s not going to stop
It’s not going to stop
It’s not going to stop ’til you wise up

No it’s not going to stop ’til you wise up
Now its not going to stop
So just give up

(lyrics by Amie Mann - from the movie Magnolia)

Chapter 1

Introduction

Information technology is often used to help people with storing and finding information. Search systems like those used in libraries or on the Internet are often centralized, which means that all search queries are being processed by one big computer. Such a system often contains a manually maintained index, like in many libraries, or builds this index via an automated crawling process combined with a statistical indexer, like Excite or Google. The big advantage of a centralized search engine, in contrast to a decentralized one, is that almost no network traffic is needed to get the answers on the queries. When the search-engine is quick, the answers are shown to the user often within a second. So, in many situations users are very satisfied with those centralized solutions. However, there are some disadvantages, which sometimes become very important. A recent example is that a search engine helped with facilitating censorship for a large Asian country. Via this way, the government was able to influence which results are shown to the users of the system. Also when censorship is not applied, the search engine is in full power to decide which information is shown and in which order. By this, companies can pay the search-engine to give their information a higher rank then their competitors in the result list. Obviously, this order does not always reflect the needs of the user. Another disadvantage of the centralized approach is the possibility to infringe privacy of the users. Every computer often maintains the same unique IP-address, which makes it easy to keep track of all queries that are posted from each machine.

These disadvantages are enough motivation to take a look at the obvious alternative: decentralized search systems. A pure decentralized search system consists of a network of connected computers where every member has the same functionality, which means that there is no hierarchical organization. In those systems, search-queries (initiated by a user on a computer somewhere in the network) are sent to some other computers which hopefully are able to fulfill the request of the initiator. This search

process normally stops after a fixed number of messages or steps or when the user indicates that (s)he is satisfied. The flat organizational structure leads to an unpredictable route of the search queries, which makes it much more difficult to censor information because every member of the network only can control its own response to queries. Also privacy is better protected because only a fraction, if any, of the queries of an individual is posted to another, possibly malicious, individual in the network.

The success of semi-decentralized systems is already shown by the popularity of file-sharing systems like Napster, BitTorrent and KaZaa, and the instant messaging systems like MSN-Messenger. These Peer-to-Peer (P2P) systems have the convenient property that the important part of network load and storage space is shared over the members of the network. As the word already indicates, semi-decentralized systems still have some hierarchical organization or centralized component. In those systems, sending and receiving data (which consumes most of the bandwidth) is done directly from provider to consumer, however finding a provider for a request from a consumer is often done via a centralized matchmaking system. In other words, these semi-centralized systems still suffer from the two disadvantages of censorship and privacy problems, but solve the (also important) scalability problem.

Pure decentralized systems are mostly still in a research stadium, where FreeNet and GNUtella are some positive exceptions that escaped the laboratory. The big challenge for the pure decentralized approach is to find peers, without using a central index, that have matching content on a query posted by a member somewhere in the network. In this thesis, three publications are bundled which all describe a specific solution to this problem. They all are based on the so-called ”Expertise-based selection model” which is introduced in the first paper. In this model peers summarize their content in expertise descriptions and advertise them to other peers. The result is that peers are going to know some peers not only by address but also by their expertise and thereby form a so-called ”Semantic Overlay Network”. The word ’semantic’ means something like a human and machine readable description of some domain of interest. The big advantage of having knowledge about the content of other peers is that queries can be forwarded to the best matching ones (instead of random), resulting in a more efficient forwarding process, i.e. less bandwidth usage.

This leads to the following research question...

1.1 Research Question

The goal of this work is to contribute to make completely distributed content search systems a good alternative for (semi)-centralized systems.

In order to realize this goal, the following research question is formalized:

How can semantic descriptions of peers in a pure decentralized Peer-to-Peer system help to improve the efficiency of the query routing process?

1.2 Contributions

Via this thesis we hope to contribute in the following ways:

1 Providing a *generic model* for a P2P system in which the routing of query messages is based on *advertised semantic descriptions* of the content from peers.

2 Developing a *routing method* and providing *simulation and field-study results* of a P2P system in which peers describe their content by topics from a *manually built shared ontology*.

3 Describing a *routing method* for peers to describe their content by terms from an *automatically built term similarity matrix*.

4 Developing a *routing method* and *providing simulation results* of a P2P system in which peers describe their content by terms from the previously mentioned automatically built term similarity matrix.

5 Developing a *routing method* and providing *simulation results* of a P2P system that combines two existing routing approaches: *Semantic Overlay Networks* and *Distributed Hash Tables*.

6 Generating several *large and realistic data-sets* which are used in the simulations, but are also useful for researchers that want to do simulations on P2P systems.

1.3 Outline of thesis

The main content of this thesis is organized in three sections, where each of them describe one of the three routing systems, together with their simulation- and field experiments, results and conclusions.

1.3.1 (Chapter 2) Bibster: expertise-based peer selection in Peer-to-Peer networks

In this chapter, the *Expertise-based selection* model is introduced. In this model, implemented as the routing algorithm in Bibster [Haase et al., 2004a], peers describe the information that they share on the network in *expertise descriptions* and their search queries in *query abstractions*. The expertise descriptions are summaries and/or abstractions of the information that they share. For example, when a peer shares some text documents, the expertise description could contain the most interesting terms. These descriptions are advertised to some peers, so that they know about the 'expertise' of the advertiser. This information about peers is used in the query forwarding process: when a peer receives a query, it selects, besides trying to match its own content, a set of peers whose expertise descriptions are the best match with the query abstraction. In this way the forwarding process should be more efficient than random query forwarding. The model is schematically presented in Fig. 1.1.

In the original paper the model prescribes that peers describe their expertise and query abstractions in terms of a shared ontology, however in chapter 3 and 4 the proposed systems drop this assumption. An ontology is a formal, explicit specification of a shared conceptualization [Gruber, 1993], mostly in the form of a semantic network. In this chapter the model is instantiated by a bibliographic scenario where the ontology is instantiated by the ACM topic hierarchy [acm,], which is a collection of topics from the computer science domain, organized by `subtopic` and `seeAlso` relations. For example, *Object Oriented Programming* is a `subtopic` of *Programming Techniques*. This means that the peers describe their 'expertise' in a subset of topics from this topic-hierarchy. [Li et al., 2003] describes different distance measures for semantic networks for determining the semantic relatedness between two topics. The authors compare it with a data-set that contains the results of a questionnaire from a group of people that gave the semantic relatedness between a set of topics. For our simulations we choose the distance measure of which the results have the highest correlation with the given data-set. Via this relatedness function, peers are able to calculate the 'closest' advertisement matches for queries which enables them to forward queries to the most relevant peers. In our bibliographic scenario we simulate different distributions of content, namely 'topic-distributions' and 'conference-distributions'. In means that in the first distribution peers are specialized on certain topics and in the second case peers contain documents from certain conferences/journals. In our field experiment the distribution of content is the set of BibTex items that a user shares on the network.

The results of our simulation experiments and field-study show that we can

significantly reduce the number of query messages that are needed to get a certain recall of the relevant documents in the network, compared to a system that forwards queries on a random basis (like in Gnutella). The number of messages needed to build the semantic overlay via advertisements, is only a fraction of the number of the reduced query messages.

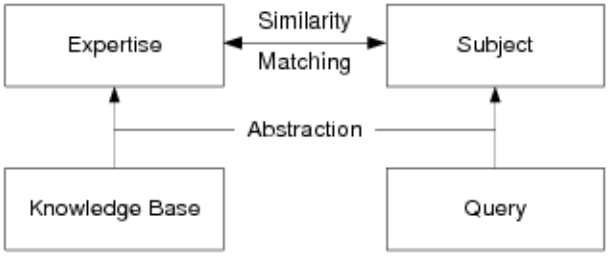

Figure 1.1: Expertise Based Matching

1.3.2 (Chapter 3) pRoute: peer selection using shared term similarity matrices

In chapter 3 we describe the pRoute approach, where the shared ontology of the expertise-based selection model of chapter 2 is replaced by an automatically generated 'term similarity matrix'. The advantage is therefore, that the development of the data-structure is done automatically and therefore consumes less effort of the users. The term similarity matrix contains information about the semantic relatedness between a large number of topics, where the relatedness can be interpreted in a similar way as described in chapter 2. The terms in the matrix are originating from an extraction process by an NLP (Natural Language Processing) application [Maedche and Staab, 2001]. This application selects, for a given set of documents, those terms which are typical for them via filtering of stop-words, stemming and looking at occurrence frequencies. The distance matrix itself is generated by a technique which is called "Latent Semantic Indexing" [Deerwester et al., 1990]. LSI transforms a document $\times$ term matrix to a term $\times$ term matrix via a statistical analysis on the correlations between terms. The 'semantic relatedness' of terms is therefore assumed to be measurable via correlations of occurrences of those terms.

Such a matrix is distributed among the members in the P2P network, assuming that the terms from the matrix are sufficient to express the expertise of the individuals. To make this assumption realistic, it is important to select a representative set of documents that are typical for the group of peers that are in the network.

Further on in the chapter we give a set of advertisement- and query policies that allow peers to make decisions about when to send and accept advertisements and to select peers to send queries to. These policies are simulated on an efficient and modular simulation platform. The data-set which is used in the simulations are the research articles of more than 100.000 researchers from the computer science domain. The results of the simulations show that the performance of the system is similar to the routing protocol as used in the Bibster system from the previous chapter.

1.3.3 (Chapter 4) pNear: combining content clustering and distributed hash tables

In chapter 2 and 3, we described two examples of Semantic Overlay Networks, that make use of a shared data-structure in which peers describe their queries and expertise descriptions. A problem of this approach is that peers which are related by a certain domain need the ontology or term similarity matrix on the domain. This data-structure has therefore to be downloaded somewhere or to be shared among the peers themselves. This results in a significant amount of network traffic, especially when topics in domain frequently undergo updates and other changes.

In this chapter we describe an alternative SON approach, which we call pNear, where peers do not describe their expertise in a shared data-structure but in their own local set of (locally extracted) terms. This approach is inspired by an approach which is based on random walk clustering, where peers with similar content are going to know each other [Voulgaris et al., 2004]. The assumption there is that queries posted by (the users of) peers are semantically closely related to the content of the peer itself. This results in a high probability that the neighbors of the peer (the peers in the cluster of that peer) have answers to the query. This original approach has two disadvantages. The first problem lies in the domain of full-text searches: the complete set of terms occurring in a peer's document-set has to be shared with other peers to allow them to determine if the sender has relevant documents for future received query terms. In other words, when there is no shared data-structure (like a fixed set of terms) in which they can describe their content, the whole content has to be shared. This results in the fact that much data has to be shared between peers for determining closeness during the random walk process. Thus in a dynamic network with many peers leaving and joining, and/or where content is very often changing, many random walk procedures will be started, resulting in much network consumption when peers exchange their descriptions. Another problem is that sometimes peers are not able or want to share content, which means that they cannot be clustered. When those peers would post a query, its neighbors would be random queries, which leads to an inefficient query

forwarding process.

In the approach in chapter 4 we try to solve the first problem of the many random walk messages by describing a faster way to cluster. We also try to tackle the second problem by providing a method for unclustered peers to efficiently find the relevant cluster during the query process. The way we do this is to combine the SON approach with another technique used for efficient query routing: Distributed Hash Tables (DHTs). DHTs are currently seen as a good competitor with the SON approach. The generic idea of DHTs is that each item that is shared on the network is hashed to a unique hash-key. This key serves as message identifier, and the message is efficiently routed to the peer whose network identifier lies closest to the key. With efficient, we mean that only $O(\log N)$ messages are needed to route the message to that peer, where N is the number of peers in the network. This means that each peer is responsible for a key-space and therefore becomes a kind of 'yellow-page' for content for this key-space. The items (and the appurtenant content) can also efficiently be retrieved if the requester knows the key. In the pNear approach, the DHT functionality is used to enable peers to register their expertise at some nodes, by selecting a small subset of the terms in their expertise descriptions which are hashed and serve as hash-key (i.e. message identifier). Via this way, a peer which is responsible (i.e. is the register) for a given (hashed) term, can receive register messages from peers that have this (hashed) term, which means that the receiver knows the senders expertise description.

These registers are used by the clustering algorithm and by the query algorithm. First, in the clustering process, a peer consults some registers which are responsible for the terms in the expertise description of that peer. These registers return the pointers it has to peers that have registered themselves for the term and therefore the consulting peer gets pointers to related peers (i.e. to peers in the relevant clusters). The clustering procedure than resumes by advertising expertise to the pointers and asking these pointers for more relevant peers. The registers are also used in the query process. The terms in the query are hashed, and the responsible registers for those terms have pointers to peers that have registered themselves on these terms. A querying peer queries the pointers, which are returned by the registers. The queried peers return besides possible matching answers also pointers to peers that also are relevant for the query (it is reasonable to assume that the queried peer knows relevant peers, because they are in the same expertise cluster).

To verify if it is true that the combination of DHT and SON solves some of the problems that uniquely hold for the single approaches, we wrote a simulation platform which is able to test different relevant parameters in the approach. The results show that we get at least the same search efficiency as DHT, however without having to store all the keys in the expertise descriptions. In the chapter we give reasonable arguments why this also reduces

problems that uniquely hold for the DHT approach.

Chapter 2

Bibster

This chapter is based on the following papers: [Ehrig et al., 2003b], [Broekstra et al., 2004], [Ehrig et al., 2003c], [Haase et al., 2004c], [Siebes et al., 2006]

Peer-to-Peer systems have proven to be an effective way of sharing data. Modern protocols are able to efficiently route a message to a given peer. However, determining the destination peer in the first place is not always trivial.

We propose a model in which peers advertise their expertise in the Peer-to-Peer network. The knowledge about the expertise of other peers forms a semantic topology. Based on the semantic similarity between the subject of a query and the expertise of other peers, a peer can select appropriate peers to forward queries to, instead of broadcasting the query or sending it to a random set of peers. To calculate our semantic similarity measure we make the simplifying assumption that the peers share the same ontology. We evaluate the model in a bibliographic scenario, where peers share bibliographic descriptions of publications among each other. In simulation experiments complemented with a real-world field experiment we show how expertise based peer selection improves the performance of a Peer-to-Peer system with respect to precision, recall and the number of messages.

2.1 Introduction

Peer-to-Peer systems are distributed systems without centralized control or hierarchical organization, in which each node runs software with equivalent functionality. A review of the features of recent Peer-to-Peer ap-

plications yields a long list: redundant storage, permanence, selection of nearby servers, anonymity, search, authentication, and hierarchical naming. Despite this rich set of features, scalability is a significant challenge: Peer-to-Peer networks that broadcast all queries to all peers do not scale - intelligent query routing and network topologies are required to be able to route queries to a relevant subset of peers. Modern routing protocols like Chord [Stoica et al., 2001], CAN [Ratnasamy et al., 2001] and Pastry [Rowstron and Druschel, 2001] are based on Distributed Hash Tables for efficient query routing, but little effort has been made with respect to rich semantic representations of meta-data and query functionalities beyond simple keyword searches.

The Semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation [Berners-Lee et al., 2001]. In a Peer-to-Peer system, Semantic Web techniques can be used for expressing the knowledge shared by peers in a well-defined and formal way. In the simple model that we propose, peers use a shared ontology to advertise their expertise in the Peer-to-Peer network. The knowledge about the expertise of other peers forms a semantic overlay network, independent of the underlying network topology. If a peer receives a query, it can decide to forward it to peers about which it knows that their expertise is *similar* to the subject of the query. The advantage of this approach is that queries will not be forwarded to all or a random set of known peers, but only to those that have a good chance of answering it.

In this paper we instantiate the above model with a bibliographic scenario, in which researchers share bibliographic metadata about publications. We present results of both simulation experiments and a real-world field experiment.

In the evaluation using the simulation experiments of our model we show how

- the proposed model of expertise based peer selection considerably improves the performance of the Peer-to-Peer system,
- ontology-based matching with a similarity measure improves the system compared with an approach that relies on exact matches, such as a simple keyword based approach,
- the performance of the system can be improved further, if the semantic overlay network is built according to the semantic similarity of the expertise of the peers,
- a “perfect” semantic overlay network imposed on the network using global knowledge yields ideal results.

The results from the field experiment with the Bibster system validate the applicability and performance of the model for real-world systems.

In the remainder of the paper we discuss related work (Section 2), present the formal model for expertise based peer selection (Section 3), instantiate this model for the bibliographic scenario (Section 4), define evaluation criteria (Section 5), present results of the simulation experiments (Section 6) and the field experiment (Section 7), and conclude with some directions for future work (Section 8).

2.2 Background and Related Work

Peer-to-Peer systems are typically characterized by the absence of a single central instance of control. This has consequences for the network organization and the coordination to route requests to the experts able to respond to the request. Peer selection plays a role in all Peer-to-Peer systems that are dealing with document discovery. By definition, any such system must have *a* strategy for peer selection (even if it is only a trivial network strategy), and many systems try to improve on this in order to avoid network congestion.

In completely unstructured Peer-to-Peer networks, the data is distributed randomly, and broadcasting mechanisms are used to distribute queries. In structured networks networks, a distributed index is built to route search requests. This structure can involve various degrees of central coordination or global knowledge, e.g. relying on super-peers. Further, we can distinguish whether the indexing structure relies on exact (syntactic) matches of keys to route requests, or whether they consider the semantics of the request.

Although many real systems which are concerned on finding expertise make use of approaches that combine developments from different research fields, they will more or less fit or be a combination of one of the following techniques:

Broadcasting. Although a very simple technique, broadcasting has already proven its usefulness in small networks and in larger Peer-to-Peer file-sharing systems [Kan, 2001]. The idea is that peers keep forwarding a query to their neighbors until a sufficient number of answers is found or till maximum number of forwards (hops) are reached. This approach is not very scalable, because a query can result in a large number of messages which consumes an unacceptable usage of network capacity. Also it is possible that even if the data is somewhere in the network it will not be found due to the maximum number of hops. The big advantage of broadcasting approaches is that they have very low maintenance costs and dependency, meaning that almost no messages are needed to keep the network alive and that the network is very robust to frequent peer drops and joins (network dynamics). In case where broadcasting really is needed, Hypercup [Schlosser et al., 2002] guarantees that only $O(N-1)$ messages and $O(\log(N))$ hops are needed to

reach all peers, where N is the number of peers in the network. Moreover, they show how their scheme can be made even more efficient by using a global semantic network to determine the organization of peers in the graph topology. Namely, when peers describe their content in terms of this shared data-structure, peers are able to cluster themselves with similar peers. This approach based on a structured hypercube overlay has more maintenance overhead and is therefore also more sensitive to network dynamics than traditional broadcasting approaches.

Central registries. An easy but not very robust approach is to have a single register where systems can advertise their expertise descriptions or to have the registry itself search the network for expertise descriptions. A well known example from the Peer-to-Peer community, but only partially Peer-to-Peer, is Napster[1]. This system has one large repository which combines filenames with peers that offer those files for downloading. Such a repository can be seen as yellow pages, where each member in the network can look up the person or system that fulfills its needs. In small organizations, such an approach could work very well because the network is small and stable, so that the registry does not have to do much query processing and updates. In larger networks the approach is not very robust and has the same disadvantages as completely centralized approaches: undisclosed content, scalability problems, lack of privacy and censor possibilities.

Brokering. The Multi-Agent community suggested the concept of 'broker agents' like in InfoSleuth [Bayardo, Jr. et al., 1997], which semantically match information needs (specified in terms of some shared data-structure, e.g. an ontology) with currently available resources which are found by the broker itself or registered by the providing agents. In InfoSleuth, agents advertise their services to the broker via the KQML [Finin et al., 1994] language. Broker agents respond to an agent's request for service with information about the other agents that have previously advertised relevant services. The literature on broker agents has a clear focus on finding services. Therefore, it is not surprising that the brokering approach is very popular in the literature on finding web-services which are semantically described [McIlraith et al., 2001]. One thing where the literature is not clear about is on how scalable and robust this approach is. In a network where millions of agents offer their services, one broker agent probably will not be enough and will have the same problems as with a central registry.

Super Peers/nodes. An approach that looks very similar to brokering but with a different goal in mind, comes from the Peer-to-Peer research community. The technique, which works well for file sharing, makes use of

[1] Napster. http://www.napster.com/about_us.html, 2002

the different capacities of the nodes in a Peer-to-Peer network: Peers that have more processing power, memory or network bandwidth than other peers are assigned additional tasks in the network. For example, KaZaa [Leibowitz et al., 2003] lets peers voluntary act as super peers that maintain large routing tables, in which information is stored about the content of other peers (comparable to yellow pages). Relying on super peers, this approach introduces a form of centralization in the system. Although better than broadcasting in a network without super-nodes, this remains essentially broadcasting and therefore can be improved by techniques that do more efficient routing described in the next paragraphs.

[Nejdl et al., 2003] presents schema-based Peer-to-Peer networks and the use of super-peer based topologies for these networks, in which peers are organized in hypercubes. This topology guarantees that each node is queried exactly once for each query. [Löser et al., 2003] shows how this schema-based approach can be used to create Semantic Overlay Clusters in a scientific Peer-to-Peer network with a small set of meta-data attributes that describe the documents in the network. In contrast, the approach in our system is completely decentralized in the sense that it does not rely on super-peers.

Distributed Hash Tables and Distributed Search Trees. Another technique that comes from the Peer-to-Peer research community makes use of Distributed Hash Tables (DHT). DHTs are based on the idea to route content (or a pointer to the content) to the peer whose identifier lies closest to the unique identifier of the content. This technique assumes that all peers have the same 'hash' function to assign a unique (mostly 128 bit) identifier to content, which could be anything like documents, music, URLs or words. The characteristic of this technique is that it allows to route content and queries in O(log(n)) steps to the right peers, where n is the number of peers in the network. Also, systems that do routing based on DHTs, such as Chord [Stoica et al., 2001] and Pastry [Rowstron and Druschel, 2001], are robust with respect to rapid join and leaves of peers. A disadvantage of most DHT approaches is that they have high maintenance costs, due to the frequent changes in the overlay network as a result of peers continuously joining and leaving. P-Grid [Aberer, 2001] is a Peer-to-Peer search system based on a virtual distributed search tree, similarly structured as standard distributed hash tables, but with an unstructured way of building the DHT-overlay. Namely, P-Grid uses randomized algorithms for constructing the access structure, updating the data and performing search. In this way probabilistic estimates can be given for the success of search requests, and search is more robust than the previously described DHT approaches against failures of nodes. A disadvantage of all DHT approaches is that objects that are not hashed cannot be found, which is a problem for full-text searching. To be specific, in a document sharing case, one could roughly do two things: (1) The file itself is hashed to a unique key. The disadvantage is that the user

has to know this key too, which is highly unrealistic. (2) The title of the document is hashed. This is still a problem because one type error would result in a complete different hash key. (3) All the words in the document are hashed and the document or the location of the document is stored at the peers on which the identifiers are closest to the hash keys of the words. Although now someone is able to find the documents that contain the keywords, the procedure of distributing the hash keys is not efficient because all these keys have to be distributed to the right peers in the network. Another disadvantage of a pure DHT-based approach is that load-balancing is not an emergent property of the topology. Due to the fact that content and queries follow a power law distribution, some peers (responsible for popular keys) are much more loaded than other peers that accidently are responsible for less popular ones. Therefore, active load-balancing strategies have to be developed on top of DHT, which is not needed for broadcast-based and expertise-based (described in next paragraph) alike approaches. Also, a pure DHT-based approach is less robust than broadcast-based and expertise-based approaches, because normally only one peer is responsible for one key, and if that peer does not respond to queries (for example behind a fire-wall or due to overload), no content can be found that is hashed to that key. The work of Byers et al. [Byers et al., 2002] confirms the load-balancing and bottleneck problem and describes an alternative DHT approach to solve it by introducing redundancy of content pointers in the network, which however generates significant additional maintenance costs.

Semantic Overlay Networks. Peers that keep pointers to other peers which have similar content to themselves form a Semantic Overlay Network (SON). Gridvine [Aberer et al., 2004] provides semantic overlay network on top of PGrid: While PGrid as a structured Peer-to-Peer network for efficient routing of messages provides the 'physical' layer, Gridvine introduces a semantic overlay for managing and mapping data and metadata schemas as the 'logical' layer. In essence, the efficiency of the search algorithm is caused not by smart forwarding queries based on the semantic overlay, but by applying the underlying DHT approach for mapping terms to peers.

Because of the focus of our own work on semantic topologies, we look closer at systems where the goal is an efficient search mechanism based on routing queries to peers that are semantically closest to the content of the query.

One approach to achieve that is to classify the content of a peer into a shared topic vector where each element in the vector contains the relevance for that given peer for the respective topic. pSearch [Tang et al., 2002], is such an example where documents in the network are organized around their vector representations (based on modern document ranking algorithms) such that the search space for a given query is organized around related documents, achieving both efficiency and accuracy. In pSearch, for each element in the topic vector, each peer has a responsibility for a certain range or interval, e.g

$([0.2 - 0.4], [0.1 - 0.3])$. Now all expertise vectors that fall in that range are routed to that peer, meaning that, following the example vector, the expertise vector $[0.23, 0.19]$ would be routed to this peer and $[0.13, 0.19]$ not. Besides the responsibility for a vector range, a peer also knows the list of neighbors which are responsible to vector ranges close to itself. The characteristic of pSearch is that the way that peers know about close neighbors is very efficient. A disadvantage of pSearch is that all documents have to be mapped into the same (low dimensional) semantic search space and that the dimensionality on the overlay is strongly dependent of the dimensionality of the vector, with the result that each peer has to know many neighbors when the vectors have high a dimension.

Another approach is based on random walk clustering [Voulgaris et al., 2004], where peers with similar content are going to know each other. The assumption is that queries posted by (the users of) peers are semantically closely related to the content of the peer itself. This results in a high probability that the neighbors of the peer (the peers in the cluster of that peer) have answers to the query. The problem of this approach in the domain of full-text searches, is what information a peer has to tell to another peer so that they are able to determine if they are related or not. When there is no shared data-structure (like a fixed set of terms) in which they can describe their content, the whole content has to be shared. This results in the fact that much data has to be shared between peers for determining closeness.

Caching of pointers to popular content based on query answers is done in Freenet [Clarke et al., 2001]. In short, when a node forwards a request for a particular key to another node in the network, and that node is successful in retrieving the data, the address of an upstream node (possibly the one where the data originated) is included in the reply. The requester makes a note of the requested key, and the source node passed back with that reply. It is assumed that the upstream node is a good place to route future requests for keys closest to the previously requested key.

There is also work on 'routing indices' where a peer maintains knowledge about the reachable content from its neighbors. For example, the work of [Crespo and Garcia-Molina, 2002] describes a method where peers summarize their knowledge in a set of topics and advertise this with the number of documents that they can reach to their direct neighbors. With 'reaching' the authors mean that the peer itself has documents on that topic, or knows other peers that have such documents. The problem with this approach is that either these index tables are very large (resulting in expensive maintenance because these indexes are sent to neighbors when updates occur) or are not rich enough to have an overlap of tables between peers, resulting in dead-ends a forwarding process.

In contrast to the previous approach, the last SON approach that we discuss here lets peers describe their content in a shared set of terms. Mostly these

terms are organized in a topic network or hierarchy making it able to determine the semantic similarity between terms. Each peer is characterized by a set of topics that describe its expertise. A peer knows about other peer's expertise topics by analyzing advertisement messages [Haase et al., 2004c] or answers [Tempich et al., 2004]. In this way peers form clusters of semantic related expertise descriptions. Given a query, a shared distance metric allows to forward queries (described by a shared set of terms) to neighbors whose expertise description is semantically closely related to the query. The advantages of this approach are threefold:

- *Peer autonomy* Each peer can, in principle, have its own distance measure, peer selection mechanism and advertisement strategy. This allows peers, for example to keep their neighbor list or similarity metric secret. Also peers can decide at any time to change their visibility on the network by sending advertisement messages.
- *Automatic load balancing* When some content is provided by many peers also the semantic cluster on that content will contain many peers. In this way, load balancing is an emergent property of this approach.
- *Robustness/fault tolerance* When peers leave the network or do not respond to a query, the only consequence is that they probably will not be asked a next time until they send new advertisement messages or are recommended by other peers. In contrast, most DHT approaches have to move routing tables to other peers in order to restore the overlay.

However there is also a disadvantage, terms that are not shared can not be found. For example, imagine that a peer has some documents containing the phrase 'database languages', but the shared data-structure only contains the term 'databases', then two things can be done (1) extend the shared data-structure with the word 'database languages' so that peers are able to query and describe their expertise with that term or (2) the functions that extracts the expertise description and abstract the queries should be intelligent enough to see that 'databases' is a good replacement for 'database languages'. Note that in this case the original query still contains 'database languages', but the routing mechanism uses the shared term 'databases' to route it to the peer that registered itself on that term. Both solutions have their own problems, the first one will lead eventually to very large data-structures, the second one depends very heavily on the quality of the extraction and abstraction algorithms.

2.3 A Model for Expertise Based Peer Selection

In the model that we propose, peers advertise their expertise in the network. The peer selection is based on matching the subject of a query and the expertise according to their semantic similarity. Figure 2.1 below shows the

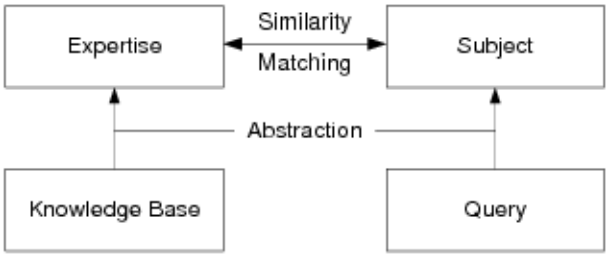

Figure 2.1: Expertise Based Matching

idea of the model in one picture. Our model is deliberately simple, in order to make as few assumptions as possible about the architecture of both the network and the individual peers, so as to make our work as widely applicable as possible.
In this section we first introduce a model to semantically describe the expertise of peers and how peers promote their expertise as advertisement messages in the network. Second, we describe how the received advertisements allow a peer to select other peers for a given query based on a semantic matching of query subjects against expertise descriptions. The third part describes how a *semantic topology* can be formed by advertising expertise.

2.3.1 Semantic Description of Expertise

Peers The Peer-to-Peer network consists of a set of peers P. Every peer $p \in P$ has a knowledge base that contains the knowledge that it wants to share.

Common Ontology The peers share an ontology O, which provides a common conceptualization of their domain by defining a set of terms and the relations between them. The ontology is used for describing the expertise of peers and the subject of queries. Although we assume that all peers share the same ontology, it can be expected that a partial overlap between different ontologies would give similar results. Distributing the ontology to all peers can be done when the user downloads the application. When the ontology is rich enough to cover most content in the network, it only needs to be downloaded once and therefore it is not a problem when the size of it would be in order of Mega-Bytes. In the next Section we use a topic hierarchy as an instantiation of the shared ontology. Topic hierarchies are relatively small things to store, in fact the topic hierarchy we use in our experiments is only 31 KB's of compressed RDF.

Expertise An expertise description $e \in E$ is an abstract, semantic description of the knowledge base of a peer based on a set of terms from the common ontology O. This expertise can either be extracted from the knowledge base automatically or specified in some other manner.

Advertisements Advertisements $A \subseteq P \times E$ are used to promote descriptions of the expertise of peers in the network. An advertisement $a \in A$ associates a peer p with an expertise description e.

Advertisement Distribution Algorithm Peers decide autonomously, without central control, whom to promote advertisements to and which advertisements to accept. This decision can be based on the semantic similarity between expertise descriptions.

2.3.2 Matching and Peer Selection

Queries Queries $q \in Q$ are posed by a user and are evaluated against the knowledge bases of the peers. First a peer evaluates the query against its local knowledge base and then decides which peers the query should be forwarded to. Query results are returned to the peer that originally initiated the query.

Subjects A subject $s \in S$ is an abstraction of a given query q expressed in a set of terms from the common ontology O. The subject can be seen a complement to an expertise description, as it specifies the required expertise to answer the query.

Similarity Function The similarity function $SF_S : S \times E \mapsto [0, 1]$ yields the semantic similarity between a subject $s \in S$ and an expertise description $e \in E$. An increasing value indicates increasing similarity. If the value is 0, s and e are not similar at all, if the value is 1, they match exactly. SF_S is used for determining to which peers a query should be forwarded. Analogously, a same kind of similarity function $SF_E : E \times E \mapsto [0, 1]$ can be defined to determine the similarity between the expertise of two peers.

Peer Selection Algorithm The peer selection algorithm returns a ranked set of peers. The rank value is equal to the similarity value provided by the similarity function.

From this set of ranked peers one can, for example, select the best n peers, or all peers whose rank value is above a certain threshold, etc.

2.3.3 Semantic Topology

The knowledge of the peers about the expertise of other peers is the basis for a semantic topology. It is important to state that this semantic topology is independent of the underlying network topology. At this point, we make no assumptions about the topology of the network.
The semantic topology can be described by the following relation:

$Knows \subseteq P \times P$, where $Knows(p_1, p_2)$ means that p_1 knows about the expertise of p_2.

The relation $Knows$ is established by the selection of which peers a peer sends its advertisements to and from which peers a peer accepts advertisement. The semantic topology in combination with the expertise based peer selection is the basis for intelligent query routing.

2.3.4 Consequences of the model

An important value of the model described above is that it dictates which design decisions must be made when equipping a Peer-to-Peer network with expertise based peer selection. These decisions are as follows:

- We must define the *ontology* as a set of terms and a set of relations between them.
- We must define *two abstraction functions*: one to abstract the contents of peers to expertise descriptions (sets of terms from the ontology), and one to abstract queries to subjects (again sets of terms from the ontology).
- We must define *two advertisement policies*: to which peers should advertisements be sent, and which advertisements should be accepted.
- We must define *two similarity functions*: one to compare subjects with expertise descriptions, and one to compare expertise descriptions with each other.
- We must define a *peer selection algorithm* to decide to which peers queries must be routed.

We believe this model to be of general value in understanding Peer-to-Peer models with semantic query routing. In the following section we will instantiate this very general model for our specific experiments.

2.4 The Bibliographic Scenario

In this section we instantiate the general model for expertise based peer selection from previous section. We use a real-life scenario for knowledge sharing in a Peer-to-Peer environment.

In the daily life of a computer scientist, one regularly has to search for publications or their correct bibliographic metadata. Currently, people do these searches with search engines like Google and CiteSeer, via university libraries or by simply asking other people that are likely to know how to obtain the desired information.

The scenario that we envision here is that researchers in a community share bibliographic metadata via a Peer-to-Peer system. The data may have been obtained from BibTeX files or from a bibliography server such as the DBLP database[2]. A similar scenario is described in [Ahlborn et al., 2002], where data providers, i.e. research institutes, form a Peer-to-Peer network which supports distributed search over all the connected metadata repositories.

We now describe the bibliographic scenario using the general model presented in the previous section.

Peers A researcher is represented by a peer $p \in P$. Each peer has an RDF [Lassila and Swick, 1999] knowledge base, which consists of a set of bibliographic metadata items that are classified according to the ACM topic hierarchy[3]. The following example shows a fragment of a sample bibliographic item based on the Semantic Web Research Community Ontology (SWRC)[4]:

```
<rdf:RDF xmlns=
 "http://www.semanticweb.org/ontologies/swrc-onto.daml#"
  xmlns:rdf ="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:acm ="http://daml.umbc.edu/ontologies/topic-ont#">
<Publication rdf:about="dblp:persons/Codd81">
 <title>The Capabilities of
        Relational Database Management Systems.</title>
 <acm:topic rdf:resource=
   "http://daml.umbc.edu/ontologies/classification#
    ACMTopic/Information_Systems/Database_Management"/>
 <!-- ... -->
</Publication>
</rdf:RDF>
```

Common Ontology The ontology O that is shared by all the peers is the ACM topic hierarchy. The topic hierarchy contains a set, T, of 1287 topics in the computer science domain and relations $(T \times T)$ between them: *SubTopic* and *seeAlso*. It is important to state that this topic hierarchy is not an 'ISA'

[2] http://dblp.uni-trier.de/

[3] http://daml.umbc.edu/ontologies/classification

[4] http://ontobroker.semanticweb.org/ontos/swrc.html

hierarchy, but a generalization/specialization organized tree structure. When it would be an ISA hierarchy, experts on a topic would also be experts on all sub-topics. This is not the case in our situation, because experts could have expertise on a very specific topic, but do not have much generic knowledge on a super-topic standing high in the hierarchy. For example, imagine an expert on *Robot Sensoring by using Bayesian Classifiers*, which is a sub-topic of *Artificial Intelligence*. This expert does not need to have any expertise on AI in general at all. This means that our topic hierarchy cannot be used for inferring expertise by inheritance over the subtopic relation. Instead, we will use to calculate a distance measure between topics.

Expertise The ACM topic hierarchy is the basis for our expertise model. Expertise E is defined as $E \subseteq 2^T$, where each $e \in E$ denotes a set of ACM topics, for which a peer provides classified instances.

Advertisements Advertisements associate peers with their expertise: $A \subseteq P \times E$. A single advertisement therefore consists of a set of ACM topics to which the peer is an expert.

Advertisement Distribution Algorithm To keep the set of simulation parameters within acceptable boundaries, we choose the simple solution of letting a peer to send its advertisement only to its direct neighbors. We therefore do not use any advertisement forwarding policy. We do however simulate different advertisement acceptance policies, which are described in one of the paragraphs from the next section on the simulation settings. The average maintenance costs of a semantic overlay can be derived by multiplying the average frequency of advertising times the average number of peers in the network.

Queries We use the RDF query language SeRQL [Broekstra and Kampman, 2004] to express queries against the RDF knowledge base of a peer. The following sample query asks for the titles of publications whose ACM topic is *Information Systems / Database Management*:

```
CONSTRUCT {pub} <swrc:title> {title} FROM
{Subject} <rdf:type> {<swrc:Publication>};
  <swrc:title> {title};
  <acm:topic>
  {<topic:ACMTopic/Information_Systems/Database_Management>}
USING NAMESPACE
swrc=<!http://www.semanticweb.org/ontologies/swrc-onto.daml#>,
rdf =<!http://www.w3.org/1999/02/22-rdf-syntax-ns#>,
acm =<!http://daml.umbc.edu/ontologies/topic-ont#>,
topic=<!http://daml.umbc.edu/ontologies/classification#>
```

Subjects Analogous to the expertise, a subject $s \in S$ is an abstraction of a query q. In our scenario, $S \subseteq 2^T$ each s is a set of ACM topics, thus $s \subseteq T$. For example, the extracted subject of the query above would be $\{$ *Information Systems/Database Management*$\}$.

Similarity Function In this scenario, we use one similarity function SF ($SF = SF_E = SF_S$), which is based on the idea that topics which are close according to their positions in the topic hierarchy are more similar than topics that have a larger distance. For example, an expert on ACM topic *Information Systems/Information Storage and Retrieval* has a higher chance of giving a correct answer on a query about *Information Systems/Database Management* than an expert on a less similar topic like *Hardware/Memory Structures*. To be able to define the similarity of a peer's expertise and a query subject, which are both represented as a set of topics, we first define the similarity for individual topics. [Li et al., 2003] have compared different similarity measures between words in WordNet, based on the hyponym relations between them. Given that the hyponym structure is a hierarchically structured generality/specificity network, we assume that this metric also applicable to our ACM topic hierarchy. Their best performing similarity measure that gave the best results on their data-set is as follows:

$$S(t_1, t_2) = \begin{cases} e^{-\alpha l} \cdot \frac{e^{\beta h} - e^{-\beta h}}{e^{\beta h} + e^{-\beta h}} & \text{if } t_1 \neq t_2, \\ 1 & \text{otherwise} \end{cases} \tag{2.1}$$

Here l is the length of the shortest path between topic t_1 and t_2 in the graph spanned by the *SubTopic* relation. h is the level in the tree of the lowest common subsumer from t_1 and t_2; $\alpha \geq 0$ and $\beta \geq 0$ are parameters scaling the contribution of shortest path length l and depth h, respectively. Based on benchmark data from [Li et al., 2003], the optimal values are: $\alpha = 0.2$, $\beta = 0.6$. Using the shortest path between two topics is a measure for similarity because Rada et al. [Rada et al., 1989] have proven that the minimum number of edges separating topics t_1 and t_2 is a metric for measuring the conceptual distance of t_1 and t_2. The intuition behind using the depth of the direct common subsumer in the calculation is that topics at upper layers of hierarchical semantic nets are more general and are semantically less similar than topics at lower levels. Our subtopic hierarchy is a tree structure, but the metric from [Li et al., 2003] is also able to deal with DAG (Directed Acyclic Graph) structures in general, by selecting the shortest path between two topics of interest.

Now that we have a function for calculating the similarity between two individual topics, we define SF as:

$$SF(s, e) = \frac{1}{|s|} \sum_{t_i \in s} \max_{t_j \in e} S(t_i, t_j) \tag{2.2}$$

This function iterates over all topics t_i of the subject s and average their similarities with the most similar topic of the expertise e.

Peer Selection Algorithm The peer selection algorithm ranks the known peers according to the similarity function described above. Therefore, peers that have an expertise more similar to that of the subject of the query will have a higher rank. From the set of ranked peers, we now only consider a selection algorithm that selects the best n peers. To prevent cycles in the forwarding loop, each query message is identified by a unique identifier and each peer only responds to each unique query only once. The costs of the algorithm in terms of the number of forwarded query messages is an experimental variable, for which the results are shown in the next sections on the simulation and field experiment.

We have now made a decision on many of the points dictated by the general model from the previous section: a common ontology, expertise and query-subject descriptions, advertisement-contents, and similarity functions. Still missing are the advertisement policy, used for propagating expertise, and the abstraction functions, used for describing content and queries. These are experimental variables because we test different policies, and therefore will be discussed in Section 6, where we describe the details of our experiments.

2.5 Evaluation Criteria

In this section we define a number of criteria for a Peer-to-Peer system, which will be the basis for the evaluation of our proposed model for peer selection. These criteria are mainly based on those described in [Ehrig et al., 2003a]. We distinguish between input parameters *affect* the performance of the system, and output parameters that *are affected* and serve as measures for the performance of the system.

2.5.1 Input parameters

The following input parameters are important criteria that influence the performance of a Peer-to-Peer system:

Number of Peers The size of the Peer-to-Peer network is represented by this number. Typically the scalability of the system is measured in terms of number of peers. The number of peers varies depending on the distribution of documents.

Number of Documents The scalability of a Peer-to-Peer system can also be expressed in terms of the number of shared resource items, e.g. documents.

Document Distribution The document distribution in Peer-to-Peer networks is rarely completely random, but often has certain properties. With this input parameter we want to evaluate how the proposed model behaves with different document distributions.

Network Topology The performance of a Peer-to-Peer system is strongly influenced by the network topology and its characteristics. Possible topologies could for example be super-peer based, star or ring-shaped, or simply a random graph.

Advertisements The advertisements are responsible for building the semantic topology. There are various variables involved, e.g. whom to send the advertisements to and which received advertisements to include based on the semantic similarity between the own expertise and that of the advertisement.

Peer Selection Algorithm The peer selection algorithm determines which peers a query should be forwarded to. This could be a naive algorithm, which simply broadcasts a query, or a more advanced one, as the proposed expertise based peer selection.

Maximum Number of Hops The maximum number of hops determines how many times a query is allowed to be forwarded. It determines how much the network will be flooded by a single query.

2.5.2 Output parameters

To evaluate a Peer-to-Peer system, we use precision and recall measures known from classical Information Retrieval. Here we distinguish measures on the document level (query answering) and the peer level (peer selection). Note that for our simulation of the bibliographic scenario we disregard the actual documents (i.e. papers) and only distribute their meta-data (i.e. their bibliographic descriptions). These measures are defined as follows:

Document level (Query Answering).

$Precision_{Doc} = \frac{|Docs_{relevant} \bigcap Docs_{returned}|}{|Docs_{returned}|}$
indicates how many of the returned documents are relevant, with $Doc_{relevant}$ being the set of relevant documents in the network, meaning that the terms in the query match their meta-data description, and $Docs_{returned}$ being the set of returned documents. We determine the set of relevant documents $Docs_{relevant}$ by evaluating the query against a centralized database which contains the complete data set. In our model we work with exact queries, therefore only relevant documents are returned. The precision will therefore always be one, meaning that the document precision is not a useful measure to use: $Precision_{Doc} = \frac{|Docs_{returned}|}{|Docs_{returned}|} = 1$.

$Recall_{Inf} = \frac{|Docs_{relevant} \bigcap Docs_{returned}|}{|Docs_{relevant}|} = \frac{|Docs_{returned}|}{|Docs_{relevant}|}$
The recall on the document level states how many of the relevant documents are returned.

Peer Level (Peer Selection).

$Precision_{Peer} = \frac{|Peers_{relevant} \bigcap Peers_{reached}|}{|Peers_{reached}|}$
For a given query, how many of the peers that were selected had relevant information. Here $Peers_{relevant}$ is the set of peers that had relevant documents and $Peers_{reached}$ is the set of peers that were reached.

$Recall_{Peer} = \frac{|Peers_{relevant} \bigcap Peers_{reached}|}{|Peers_{relevant}|} = \frac{|Peers_{reached}|}{|Peers_{relevant}|}$
indicates for a given query, how many of the peers that had relevant information were reached.

Further Parameters. Another important output parameters is:

$Number_{Messages}$
This output parameter indicates with how many messages the network is flooded by one query. The number of messages does not only affect the network traffic, but also CPU consumption, such as for the processing of the queries in the case of query messages.

There are many other output parameters that we could have used as additional evaluation criteria. Examples are the size of messages between peers, the response times on queries to the network, CPU load of individual peers etc. However, we do not report on these as they are not relevant to our evaluation hypotheses and therefore also not captured by our simulation software.

2.6 Simulation Experiments

In this section we describe the simulation of the scenario presented in Section 2.4. The evaluations are based on the criteria defined in Section 2.5. With the experiments we validate the following hypotheses:

- **H1 - Expertise based selection:** The proposed approach of expertise based peer selection yields better results than a naive approach based on random selection. The higher precision of the expertise based selection results in a higher recall of peers and documents, while reducing the number of messages per query.
- **H2 - Ontology based matching:** Using a shared ontology with a metric for semantic similarity improves the recall rate of the system compared with an approach that relies on exact matches, such as a simple keyword based approach.
- **H3 - Semantic overlay network:** The performance of the system can be improved further, if the semantic topology is built according to the semantic similarity of the expertise of the peers. This can be realized, for example, by accepting advertisements that are semantically similar to the own expertise.
- **H4 - The "Perfect" overlay network:** Perfect results in terms of precision and recall can be achieved, if the semantic overlay network coincides with a distribution of the documents according to the expertise model.

2.6.1 Setup of the Simulation Experiments

In the following we describe the setup of the simulation experiments performed: the data sets used, the distribution of the data, the simulation environment, and the individual experimental settings.

Data Set To obtain a critical mass of bibliographic data, we used the DBLP data set, which consists of metadata for 380440 publications in the computer science domain.

We have classified the publications of the DBLP data set according to the ACM topic hierarchy using a simple classification scheme based on lexical analysis: A publication is said to be about a topic, if the label of the topic occurs in the title of the publication. For example, a publication with the title "The Capabilities of Relational Database Management Systems." is classified into the topic *Database Management*. Topics with labels that are not unique (e.g. *General* is a subtopic of both *General Literature* and *Hardware*) have been excluded from the classification, because typically these labels are too general and would result in publications classified into multiple, distant

topics in the hierarchy. Obviously, this method of classification is not as precise as a sophisticated or manual classification. However, a high precision of the classification is not required for the purpose of our simulations. As a result of the classification, about one third of the DBLP publications (126247 out of 380440) have been classified, against 553 out of the 1287 ACM topics. The classified DBLP subset has been used for our simulations.

Document Distribution We have simulated and evaluated the scenario with two different distributions, which we describe in the following. Note that for the simulation of the scenario we disregard the actual documents and only distribute the bibliographic meta-data of the publications.

Topic Distribution: In the first distribution, the bibliographic meta-data are distributed according to their topic classification. There is one dedicated peer for each of the 1287 ACM topics. The distribution is directly correlated with the expertise model, each peer is an expert on exactly one ACM topic and contains all the corresponding publications. This also implies that there are peers that do not contain publications, because not all topics have classified instances.

Proceedings Distribution: In the second distribution, the bibliographic meta-data are distributed according to conference proceedings and journals in which the according publications were published. For each of the conference proceedings and journals covered in DBLP there is a dedicated peer that contains all the associated publication descriptions (in the case of the 328 journals) or inproceedings (in the case of the 2006 conference proceedings). Publications that are published neither in a journal nor in conference proceedings are contained by one separate peer. The total number of peers therefore is 2335 (=328+2006+1). With this distribution one peer can be an expert on multiple topics, as a journal or conference typically covers multiple ACM topics. Note that there is still a correlation between the distribution and the expertise, as a conference or journal typically covers a coherent set of topics.

We do not make any assumptions on how these distributions are achieved, so we see them as given in our simulations. One way to distribute content in this way is via DHT where the keys are topics or conference identifiers, so that each of them is mapped to a unique peer in the network. We already mentioned some problems with DHT approaches such as no load-balancing and single points of failures. Our experiments can be seen as a way to investigate how semantic methods can be used to mitigate some of these problems.

Simulation Environment To simulate the scenario we have developed and used a controlled, configurable Peer-to-Peer simulation environment. A single simulation experiment consists of the following sequence of operations:

1. *Setup network topology:* In the first step we create the peers with their knowledge bases according to the document distribution and arrange them in a random network topology, where every peer knows 10 random peers. We have fixed this number in our simulations to keep the number of different variable tractable, and have chosen this value to simulate a realistic sparse topology. We do not make any further assumptions about the network topology.
2. *Advertising Knowledge:* In the second step, the semantic overlay network is created. Every peer sends an advertisement of its expertise to all other peers it knows based on the overlay network. When a peer receives an advertisement, it may decide to store all or only selected advertisements, e.g. if the advertised expertise is semantically similar to its own expertise. After this step the semantic overlay network is static and will not change anymore.
3. *Query Processing:* The peers randomly initiate queries from a set of randomly created 12870 queries, 10 for each of the 1287 ACM topics. The peers first evaluate the queries against their local knowledge base and then propagate the query according to their peer selection algorithms described below.

We currently do not simulate any node drops and node joins, which would be needed to show how our system behaves in a dynamic environment. This clearly is future work. However, we can already say that the only effect of unreachable peers is that advertisement messages and query messages will not arrive. The consequence would be that other peers need to be selected, resulting in an increase of the number of messages and/or a sparser semantic overlay network, both gradually decreasing the performance of our system. We expect that the costs will remain to be low in a dynamic network, because the advertisement process does not consume many messages. This means that restoring the semantic overlay would not have a dramatic effect on the network load.

Experimental Settings In our experiments we have systematically simulated various settings with different values of input variables. In the following, we describe an interesting selected subset of the settings to prove the validity of our hypotheses.

Setting 1 In the first setting we use a naive peer selection algorithm, which selects n *random* peers from the set of peers that are known from advertisements received, but disregarding the content of the advertisement. This means that peers only have pointers to peers without knowing their expertise, so peer selection would be identical to random selection like in the Gnutella approach. In the experiments, we keep n=2 fixed in every setting, as a rather

arbitrary choice. Different values for n yield similar results, but degenerate to a sequence in the case of n=1 and to a broadcast in the case where n is the number of all known peers.

Setting 2 In the second setting we apply the expertise based selection algorithm. The *best* n (n=2) peers are selected for query forwarding. Here the peer selection algorithm only considers *exact* matches of topics, which means that a peer only is selected when its expertise description contains at least one of the topics from the query abstraction. In this setting, all advertisements are accepted.

Setting 3 In the third setting we modify the peer selection algorithm to use the ontology based similarity measure, instead of only exact matches. The peer selection only selects peers whose expertise is equally or more similar to the topics from the query abstraction than the expertise of the forwarding peer itself. This method guarantees that queries are forwarded to equal or better experts than the forwarding peer. The danger of this approach is that some of the forwarding branches get stuck in a local maximum because it does only know, if any, peers which are worse matches than itself. In this setting, all advertisements are accepted.

Setting 4 In the fourth setting we modify the peer to only accept advertisements that are semantically similar to its own expertise. The threshold for accepting advertisements was set to accept on average half of the incoming advertisements. The peer selection algorithm is identical to the previous setting, namely select peers based on the ontology based similarity measure.

Setting 5 In this setting we assume global knowledge to impose a perfect overlay network on the peer network. In this perfect overlay network the *knows* relation coincides with the ACM topic hierarchy: Every peer knows exactly those peers that are experts on the neighboring topics of its own expertise. This setting is only applicable for the distribution of the publications according to their topics, as it assumes exactly one expert per topic. A way to achieve this overlay network is via DHT, where for each key (i.e. topic) only one peer is responsible. This means that in this setting we build the semantic overlay on top of the assumed DHT overlay. Clearly, this setting suffers from some limitations as DHT like load-balancing problems in case of popular content, or unreachable content classified on a topic when the peer on the topic does not respond. In this setting, an advertisement is accepted only when the contained expertise description is similar to the receivers own expertise description, thus like in setting 4.

Setting nr.	*Peer selection method*	*Advertisement method*	*Topology*
Setting 1	random	accept all	random
Setting 2	exact match	accept all	random
Setting 3	ontology based match	accept all	random
Setting 4	ontology based match	accept similar	random
Setting 5	ontology based match	accept similar	perfect

Table 2.1: Overview of the simulation settings

The Table 2.6.1 summarizes the instantiations of the input variables for the described settings.

Simulation Results Figures 2.2 through 2.5 show the results for the different settings and distributions. The simulations have been run with a varying number of allowed hops. In the results we show the performance for a maximum of up to eight hops. Zero hops means that the query is processed locally and not forwarded. Please note that the diagrams for the number of messages per query and recall (i.e. Figures 2.3, 2.4, 2.5) present cumulative values, i.e. they include the sum of the results for *up to* n hops. The diagram for the precision (Figure 2.2) of the peer selection displays the precision for a particular number of hops.

In the following, we interpret the results of the experiments for the various settings described above with respect to our hypotheses H1 through H4.

R1 - Expertise based selection The results of Figure 2.2, Setting 1, show that the naive approach of random peer selection gives a constant low precision of 0.03% for the topic distribution and 1.3% for the proceedings distribution. This results in a fairly low recall of peers and documents despite a high number of messages, as shown in Figures 2.3, 2.5, 2.4, respectively. With the expertise based selection, either exact or similarity based matching, the precision can be improved considerably by about one order of magnitude. For example, with the expertise based selection in Setting 3, the precision of the peer selection (Figure 2.2) can be improved from 0.03% to 0.15% for the topic distribution and from 1.3% to 15% for the proceedings distribution. With the precision, also the recall of peers and documents rises (Figures 2.3, 2.5). At the same time, the number of messages per query can be reduced. The number of messages sent is influenced by two effects. The first effect is message redundancy: The more precise the peer selection, the higher is the chance of a peer receiving a query multiple times on different

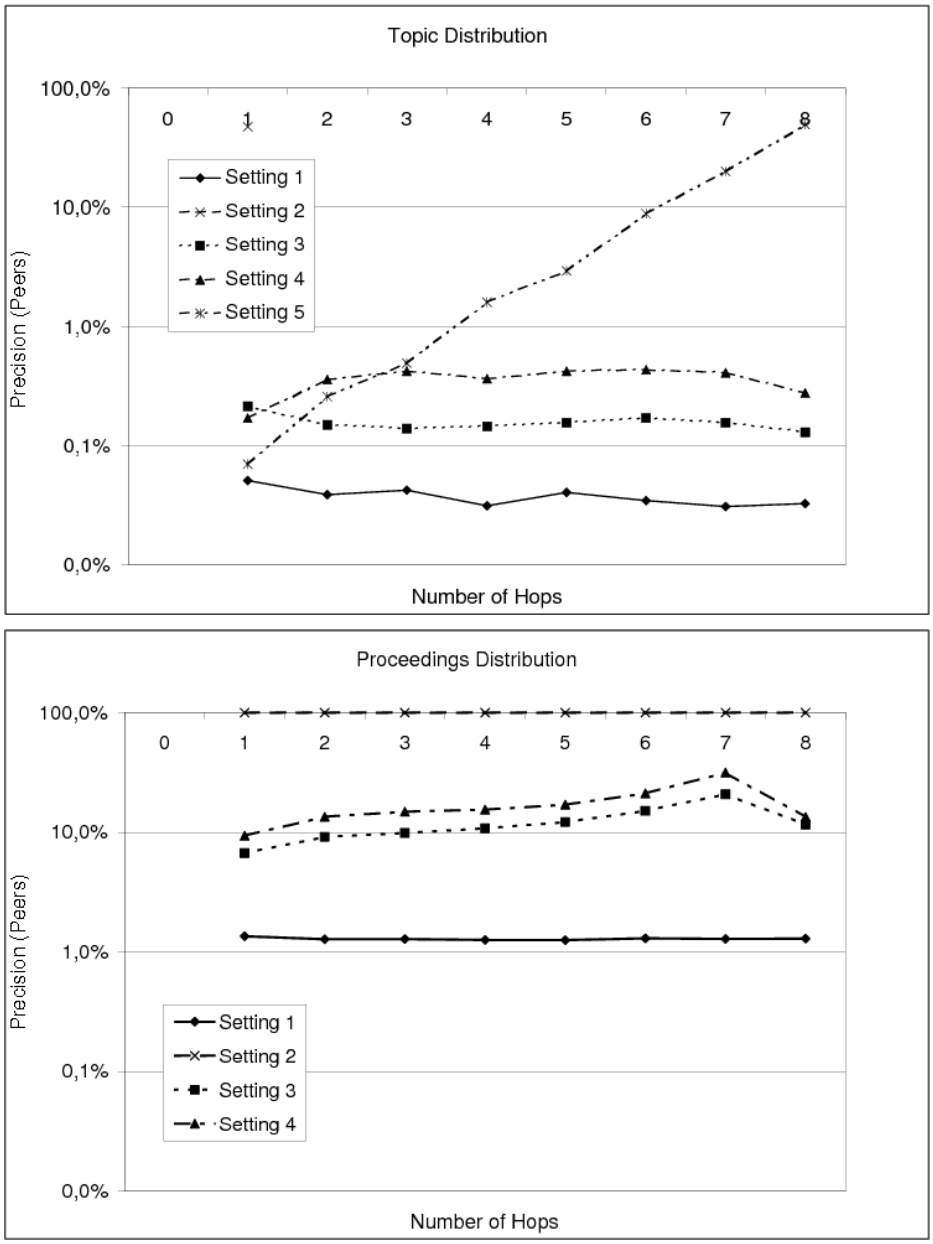

Figure 2.2: $Precision_{Peers}$

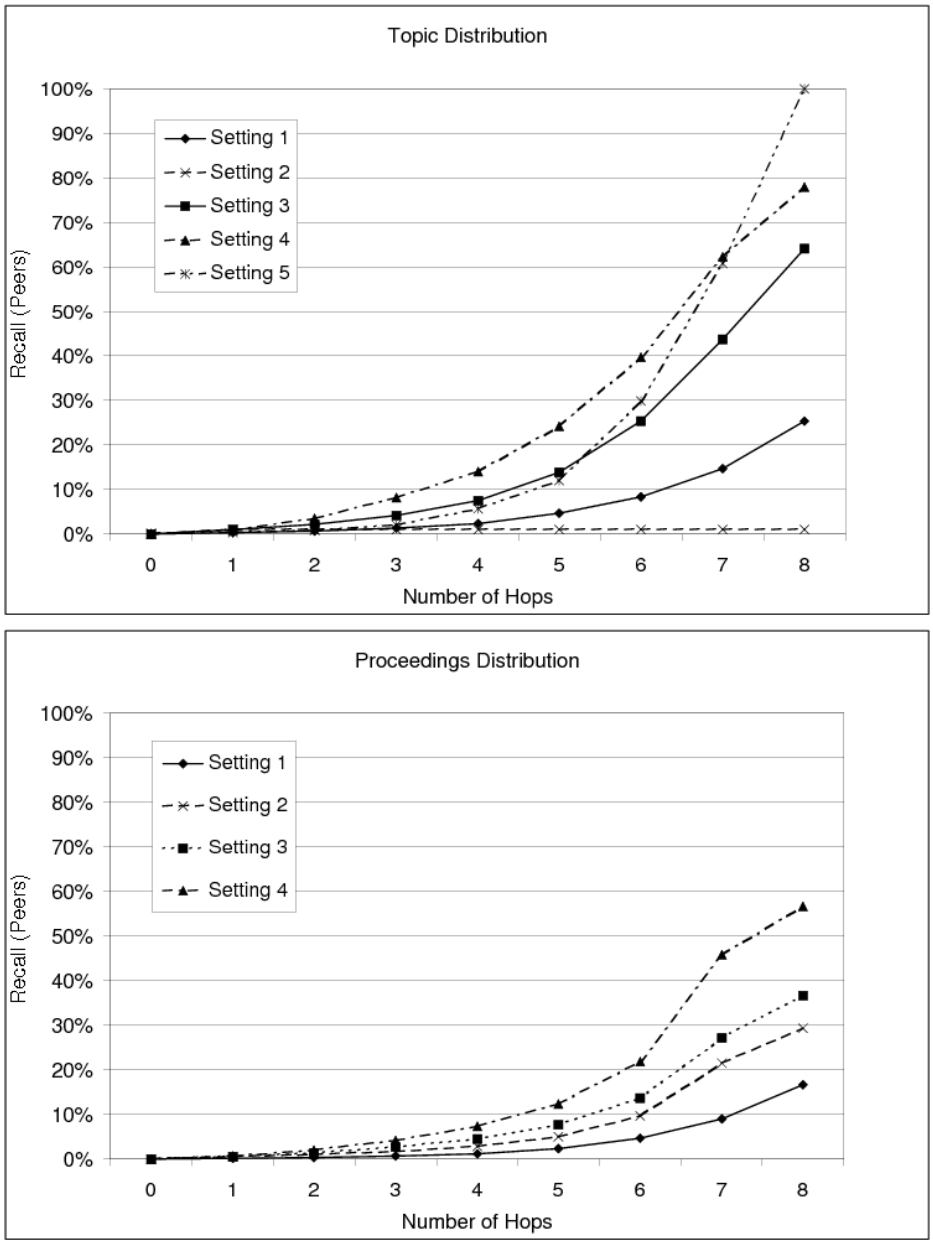

Figure 2.3: $Recall_{Peers}$

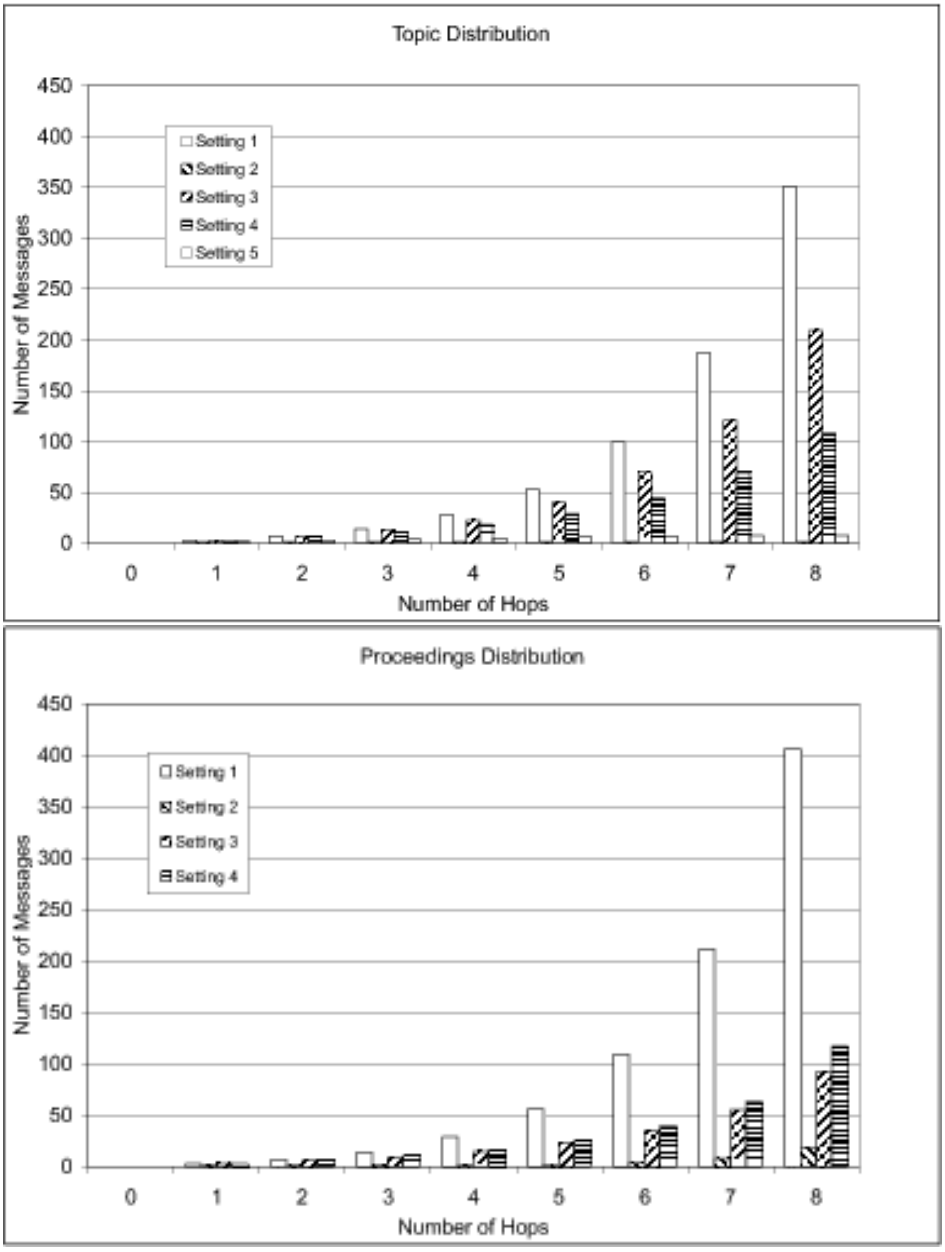

Figure 2.4: $Number_{Messages}$

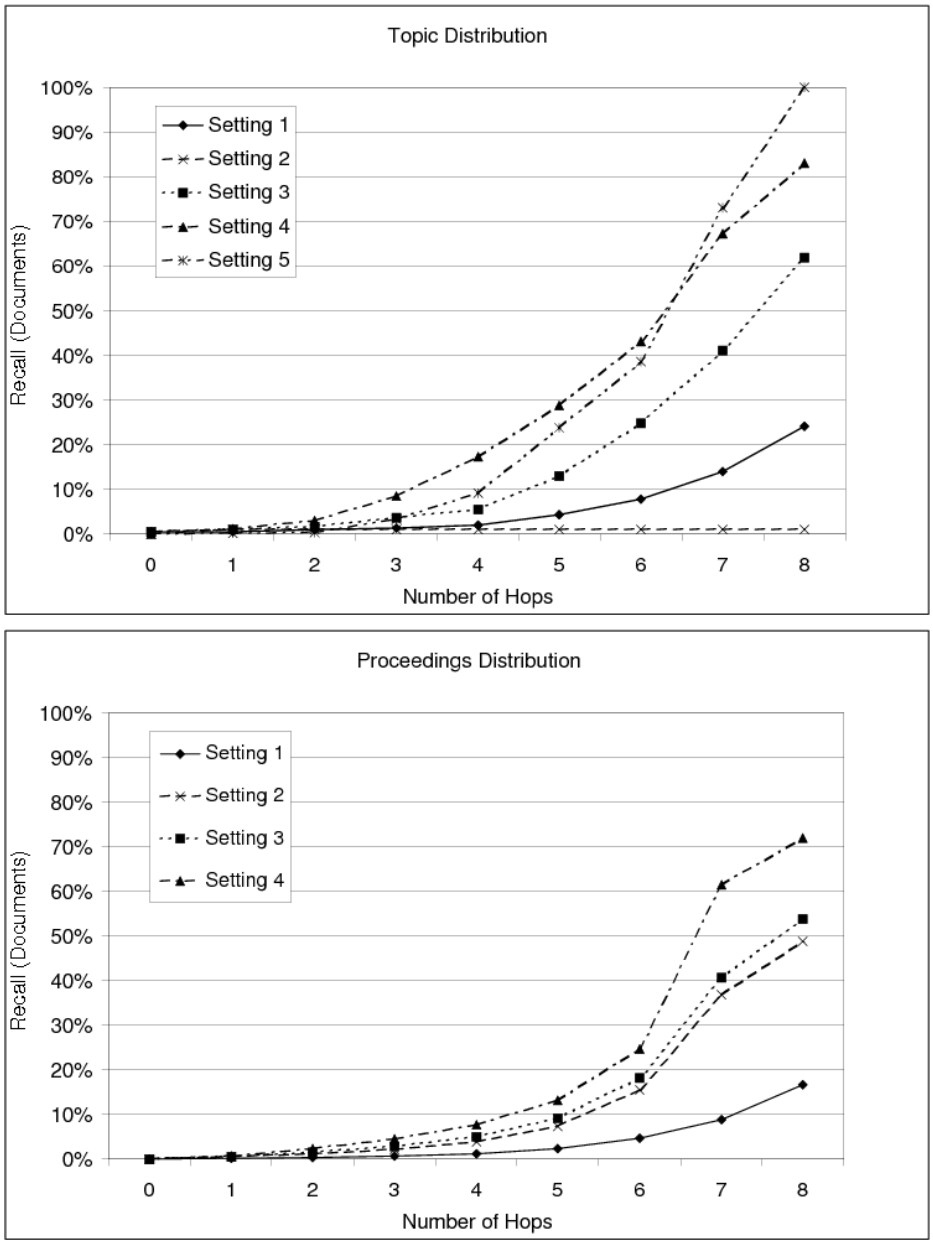

Figure 2.5: $Recall_{Documents}$

routes. This redundancy is detected by the receiving peer, which will forward the query only once, thus resulting in a decreasing number of queries sent across the network. The other effect is caused by the selectivity of the peer selection: It only forwards the query to peers whose expertise is semantically more or equally similar to the query than that of the own expertise. With an increasing number of hops, as the semantic similarity of the expertise of the peer and the query increases, the chance of knowing a qualifying peer decreases, which results in a decrease of messages.

R2 - Ontology based matching The result of Figure 2.2, Setting 2, shows that the exact match approach results in a maximum precision already after one hop, which is obvious because it only selects peers that match exactly with the query's subject. However, Figure 2.3 shows that the recall in this case is very low in the case of the topic distribution. This can be explained as follows: For every query subject, there is only one peer that exactly matches in the entire network. In a sparse overlay network, the chance of knowing that relevant peer is very low. Thus the query cannot spread effectively across the network, resulting in a document recall of only 1%. In contrary, Setting 3 shows that when semantically similar peers are selected, it is possible to improve the recall of peers and documents, to 62% after eight hops. Also in the case of the proceedings distribution, where multiple exact matches are possible, we see an improvement from 49% in the case of exact matches (Setting 2), to 54% in the case of ontology based matches (Setting 3). Naturally, this approach requires to send more messages per query and also results in a lower precision.

R3 - Semantic overlay network In Setting 4 the peers only accept semantically similar advertisements. This has proven to be a simple, but effective way for creating a semantic overlay network that correlates with the expertise of the peers. This allows to forward queries along the gradient of increasing semantic similarity. When we compare this approach with that of Setting 3, the precision of the peer selection can be improved from 0.15% to 0.4% for the topic distribution and from 14% to 20% for the proceedings distribution. The recall of documents can thus be improved from 62% to 83% for the topic distribution and from 54% to 72% for the proceedings distribution.

It is also interesting to note that the precision of the peer selection for the similarity based matching decreases slightly after seven hops (Figure 2.2). The reason is that after seven hops the majority of the relevant peers has already been reached. Thus the chance of finding relevant peers decreases, resulting in a lower precision of the peer selection.

R4 - The "perfect" overlay network The results for Setting 5 show how one could obtain the maximum recall and precision, if it were possible to impose an ideal semantic overlay network. All relevant peers and thus all bibliographic descriptions can be found in a deterministic manner, as the query is simply routed along the route which corresponds to the shortest path in the ACM topic hierarchy. At each hop the query is forwarded to exactly one peer until the relevant peer is reached. The number of messages required per query is therefore the length of the shortest path from the topic of expertise of the originating peer to that of the topic of the query subject. The precision of the peer selection increases to the maximum when arriving at the eight hop, which is the maximum possible length of a shortest path in the ACM topic hierarchy. Accordingly, the maximum number of messages (Figure 2.4) required is also eight.

2.7 The Bibster Field Experiment

In addition to the simulation experiments, we have evaluated the methods of expertise-based peer selection in a realistic field-experiment, as part of the Bibster system. The Bibster system[5] [Haase et al., 2004b] was developed as part of the EU-funded SWAP project, with contributions by many of the project team. We have implemented the methods for expertise-based peer selection in the Bibster system, and performed a public field experiment to evaluate the model in a real world setting. We are aware that the data obtained in the field experiment does not allow to make statements about statistical significant. It therefore should be seen as an addition to our simulation results and a case-study for a real life deployment.

The Bibster System Bibster is a Peer-to-Peer system for exchanging bibliographic data among researchers. Bibster exploits ontologies in data storage, query formulation, query routing and answer presentation: When bibliographic entries are made available for use in Bibster, they are structured via the SWRC ontology and classified according to the ACM topic hierarchy, both earlier mentioned in this paper. This ontological structure is then exploited to help users formulate their queries. Subsequently, the ontologies are used to improve query routing across the Peer-to-Peer network. Finally, the ontologies are used to post-process the returned answers in order to do duplicate detection. Bibster is a fully implemented open source solution built on top of the JXTA platform.

[5] `http://bibster.semanticweb.org`

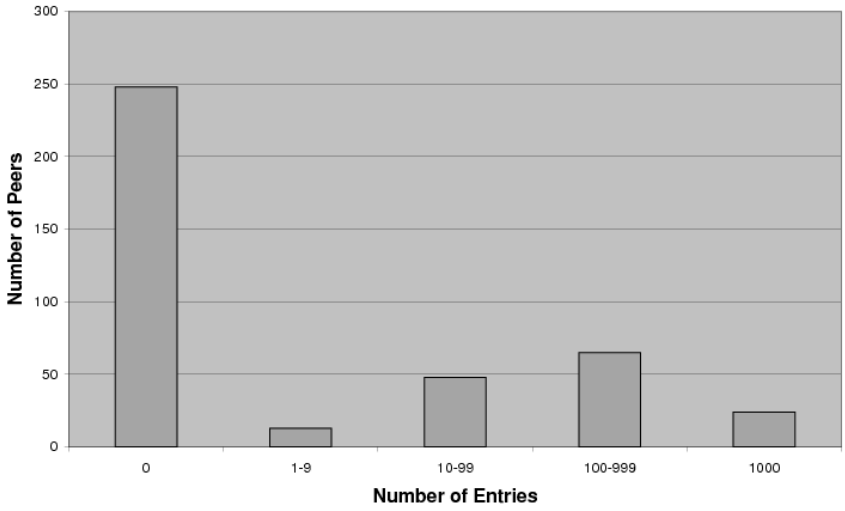

Figure 2.6: Distribution of Shared Content

2.7.1 Setup of the Field Experiment

The Bibster system was made publicly available and advertised to researchers in the Computer Science domain. The evaluation was based on the analysis of system activity that was automatically logged to log files on the individual Bibster clients. In Bibster two different peer selection algorithms ran at the same time, namely our expertise-based peer selection and a random query forwarding algorithm. We have analyzed the results for a period of three months (June - August 2004).

398 peers spread across multiple organizations mainly from Europe and North America participated in the field experiment and used the Bibster system.

A total of 98872 bibliographic entries were shared by the 398 peers, with an average of 248 entries per peer. However, the distribution had a high variance (c.f. Figure 2.6): While 62% (248 peers) were free-riding[6] and shared

[6]In many Peer-To-Peer systems (e.g. Napster, Gnutella) users are mainly interested in their own advantage and conserve their resources (i.e. bandwidth) by sharing no files. In the common literature this phenomena is called *Free-Rider* problem. Users do not have a direct incentive to share files.

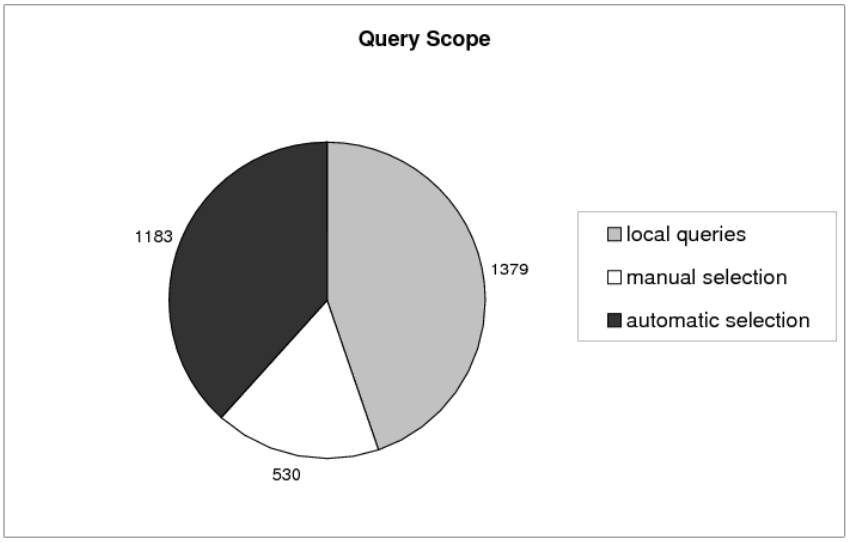

Figure 2.7: Scope of Queries

no content, 6% (24 peers) shared at least 1000 entries each, accounting for 79% of the total shared content. With respect to the variance, the distribution is similar to that of the *topic distribution* from the simulation experiments, where many peers provided no entries (those whose topic had no classified instances) and few peers provided many entries (those with popular topics such as "Database Management"). The users performed a total of 3319 queries. With respect to the scope of the queries, Figure 2.7 shows that the users mainly performed queries on their local peers and automatic search across the entire network. Only in a few cases the queries were directed to a manually selected peer. This confirms the need for efficient peer selection algorithms. For the 3319 queries, the users received a total of 36960 result entries, i.e. around 11 result entries per query. Result entries were actively used 801 times, i.e. copied or stored locally.

2.7.2 Results

With respect to query routing and the use of the expertise based peer selection, we were able to reduce the number of query messages by more than

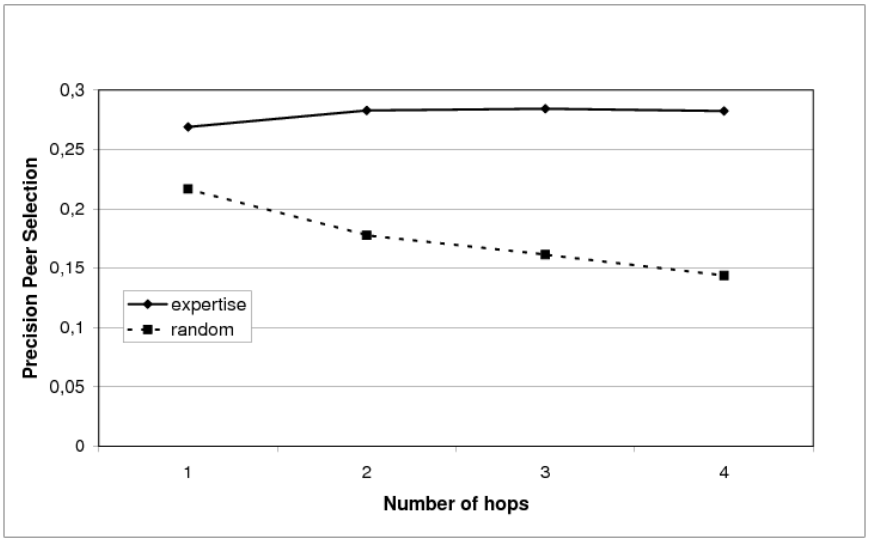

Figure 2.8: $Precision_{Peers}$

50%, while retaining the same recall of documents compared with a naive broadcasting approach. Figure 2.8 shows the precision of the peer selection (the percentage of the reached peers that actually provided answers to a given query): While the expertise based peer selection results in an almost constant high precision of 28%, the naive algorithm results in a lower precision decreasing from 22% after 1 hop to 14% after 4 hops[7].

Figure 2.9 shows the number of forwarded query messages sent per query. It can be seen that with an increasing number of hops, the number of messages sent with the expertise based peer selection is considerably lower than with the naive algorithm. Although we have shown an improvement in the performance, the results also show that with a network of the size as in the field experiment, a naive approach is also acceptable. On the other hand, with a growing number of peers, query routing and peer selection becomes critical. In the previous discussed simulation experiments, networks with thousands of peers improve in the order of one magnitude in terms of recall of documents and relevant peers.

[7]The decrease is due the redundancy of relevant peers found on different message paths: Only distinct relevant peers are considered.

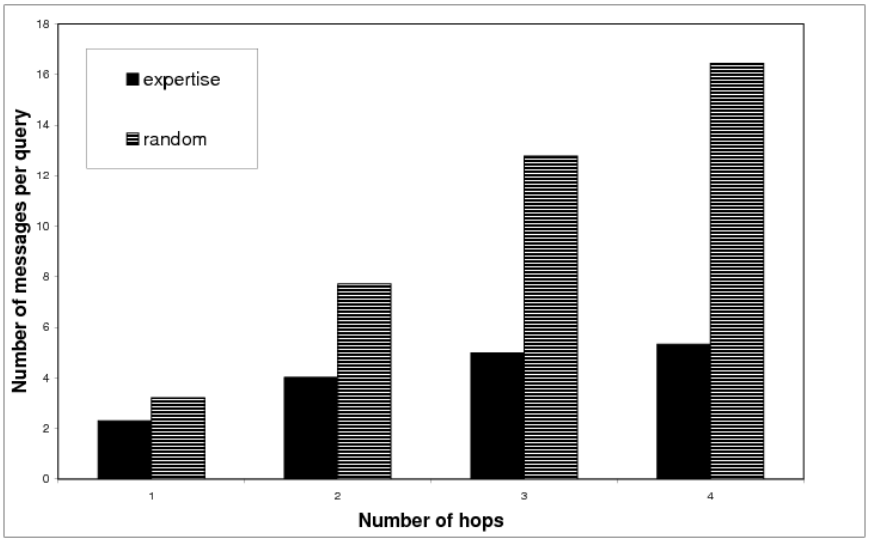

Figure 2.9: $Number_{Messages}$

2.7.3 Comparison with Results from Simulation Experiments

Overall, the results of the simulation experiments have been validated: We were able to improve the precision of the peer selection and thus reduce the number of sent messages. However, the performance gain by using the expertise based peer selection was not as significant as in the simulation experiments[8].

This is mainly due to the following reasons:

- *Size of the network* The size of the network in the field experiment was considerably *smaller* than in the simulation experiments. While the total number of participating peers was already fairly large (398), the number of peers online at any point in time was fairly small (order of tens).
- *Network topology* In the field experiment we built the semantic topology on-top of the JXTA network topology. Again, because of the small

[8]In terms of recall, there were no improvements at all, as even the naive algorithm generally was able to reach all relevant peers.

size of the network, the JXTA topology degenerates to a fully connected graph in most cases. Obviously, for these topologies, a naive algorithm yields acceptable results.

- *Distribution of the content* In the simulation experiments, we distributed the shared content according to certain assumptions (based on topics, conferences, journals). In real world experiments, the distribution is much more heterogeneous, both in terms of the expertise of the peers and the amount of shared content.

2.8 Conclusions and Future Work

In this paper we have presented a model for expertise-based peer selection, in which a semantic topology among the peers is created by advertising the expertise of the peers. We have shown how the model can be applied in a bibliographic scenario. Simulation experiments that we performed with this bibliographic scenario show the following results:

- Using expertise-based peer selection can increase the performance of the peer selection by an order of magnitude (result R1).
- However, if expertise-based peer selection uses simple exact matching, the recall drops to unacceptable levels. It is necessary to use an ontology-based similarity measure as the basis for expertise-based matching (result R2).
- An advertising strategy where peers only accept advertisements which are semantically close to their own profile (i.e. that are in their semantic neighborhood) is a simple and effective way of creating a semantic topology. This semantic topology allows to forward queries along the gradient of increasing semantic similarity (result R3).
- The above results depend on how closely the semantic topology of the network mirrors the structure of the ontology. All relevant performance measures reach their optimal value when the network is organized exactly according to the structure of the topology (result R4). Although this situation is idealized and will in practice not be achievable, the experiment serves to confirm our intuitions on this.

Also, the field experiment showed that we were able to improve the precision of the peer selection and thus reduce the number of sent messages. However, the performance gained by using the expertise based peer selection was not as significant as in the simulation experiments. Summarizing, in both the simulation experiments and the field experiments, we have shown that expertise-based peer selection combined with ontology-based matching outperforms both random peer selection and selection based on exact matches,

and that this performance increase grows when the semantic topologies more closely mirrors the domain ontology.

We have made a number of simplifying assumptions in our experiments, such as the assumption that all peers agree on the use of a single ontology, which is not realistic in all cases. We already have work in progress which allows us to relax this constraint. We expect that differences in ontologies used by different peers will *lower* our results, since the computation of the semantic distance between peers becomes less reliable across different ontologies. Currently we are working on an approach where expertise descriptions are not described in terms from a global shared ontology. Instead, routing is based on overlap between sets of locally extracted terms.

In our simulation experiments, the semantic topology was determined once, during an initial advertising round, and was not adapted any further during the lifetime of one experiment. In our field experiment this assumption was not made and also the work in [Tempich et al., 2004] shows how the topology can be adjusted based on the exchange of queries and answers. More research has to be done to show that such a self-adjusting network will *improve* the results. We think this will be the case since the semantic topology will converge better towards the structure of the underlying ontology than our current one-shot advertising allows. Currently we submitted a paper containing results on simulations with a network where content is distributed dynamically and peers update and re-advertise their expertise descriptions. In that paper we also used a more complex expertise models based on Latent Semantic Indexing.

The expertise model presented for the bibliographic scenario used in our simulations experiments is a fairly simple one, based on the ACM topic hierarchy. Other domains may require more complex expertise models with different similarity functions. One option would be, for example, to extend the expertise model with quantitative measures to indicate how much information for a certain topic of expertise is available on the peer. Another option, on which we are currently working, is to automatically extract a shared term similarity matrix based on a subset of documents retrieved from the network.

Chapter 3

pRoute

This chapter is based on the following papers: [Siebes and Kotoulas, 2006], [Siebes and Kotoulas, 2005]

Peer-to-Peer systems have proven to be an effective way of sharing data. The focus of this paper is on distributed search based on Peer-to-Peer technology. In this paper we present the pRoute system where peers advertise a short description of the content that they share, namely a set of terms. Peers remember the advertisements of related peers and thereby form a semantic overlay by which we mean that peers with similar content are grouped together. Peers calculate the similarity between their content descriptions by a term similarity function which, in the ideal case, is identical for all peers. In simulation experiments we compare the performance of different advertisement- and forwarding policies with respect to precision, recall and the number of messages. The results indicate precision and recall increase when the policies take semantics into account, without an increase of the number of advertisement- and query messages.

3.1 Introduction

Undisclosed content, lack of privacy and the possibility to censor data are seen as important disadvantages of the centralized approach of today's popular search engines. Firstly, in such a centralized approach, the owner of the server has complete control over which content and in which order the content is presented to the user. The drawback of this approach is that authorities could force the search engine not to show some content that they do not like. Secondly, much data on the web is dynamically generated via

databases and therefore are very difficult to crawl by search engines. Often this is by intention of the provider because it wants a unique access point for users to find its data, guaranteeing user traffic to the web-site and/or keeping full control over the data. Thirdly, also privacy is an important issue that is in principle not guaranteed by centralized search engines. Namely, search engines easily can associate IP-addresses with queries and can make a profile of the users behind it. Peer-to-Peer systems, where nobody is in control, are in principal much more difficult to be used for tracing the behavior of users. These three issues are important reasons for doing research on P2P-based search engines.

A big advantage of centralized search engines is that the number of messages needed in the query process often is only 2 and the number of hops is only 1, guaranteeing efficient bandwidth usage and quick response times. In Peer-to-Peer systems, the number of messages and hops mainly depends on how quickly the relevant peers are found. Needless to say that much of the current research on P2P-based document searching is focused on reducing the number of messages and hops to generate an attractive alternative to the centralized approach especially when privacy, undisclosed content and censorship play a role.

In our short literature study we try to give an impression of the different distributed search approaches and indicate respectively their strengths and weaknesses. One of the approaches described in this overview is peer selection based on semantic overlay networks (SONs), which will also be the way we organize our P2P network. More precisely, we adopt the model of expertise-based peer selection using a shared data-model as is described in the work of [Haase et al., 2004c]. In this model, peers (automatically) summarize their content in so-called *expertise-descriptions*, being a set of terms provided a shared data-model. These descriptions are spread to some other peers in the network via *advertisement messages*. In this way, peers become aware of the expertise of other peers, enabling them to route queries only to those peers whose expertise is semantically close related to the content of the queries.

By this paper we present our pRoute system and hope to contribute in the following ways: first we propose a method to automatically generate a similarity matrix that can be used as an instantiation of the shared term similarity function needed by our expertise-based routing approach. Second, we compare different query and advertisement policies via simulation results based on large peer-sets and realistic data-sets. The results should be interpreted as guidelines for those who want to build a system that uses expertise-based peer selection for routing query messages. Important to note is that the routing method can be easily combined with other techniques like using Super-peers or Distributed Hash Tables.

The structure of this paper is as follows: first we discuss related work in section 3.2. Section 3.3 describes expertise-based routing model and the

different query and advertisement policies. Section 3.4 proposes a method for automatically making a term similarity matrix which can be used as an instantiation of the shared term similarity function. Section 3.5 is about our experimental setup where the data set, the research questions and the simulation are described. Section 3.6 shows the results of our experiments and finally, section 3.7 interprets the results and indicates our future work.

3.2 Related work on distributed search

In this related work section, we organize the different systems found in the literature by methods that do not exclude each-other, meaning that a system can combine two or more methods together. Due to our focus on reducing messages and hops and increasing recall, we judge these systems on criteria concerning efficiency (network band-width, memory and cpu usages) and robustness (fault-tolerance, network dynamics).

Broadcasting Although a very simple technique, broadcasting has already proven its usefulness in small networks and in larger P2P file-sharing systems like Gnutella[1] [Kan, 2001]. The idea is that peers keep forwarding a query to their neighbors until a sufficient number of answers has been found or until the maximum number of forwards (hops) has been reached. This approach is not very scalable, because a query can result in many messages which consumes much network capacity. Thus, finding the expert that can answer a given query results in a blind search, where it is possible that even if the data is somewhere in the network, it will not be found due to the maximum number of hops. The big advantage of broadcasting approaches is that they have very low maintenance costs and dependency, meaning that almost no messages are needed to keep the network alive and it is very robust to frequent peer drops and joins (network dynamics). In case where broadcasting really is needed, Hypercup [Schlosser et al., 2002] guarantees that only $N-1$ messages are needed to reach all peers and $O(\log(N))$ hops, where N is the number of peers in the network. Moreover, they show how their scheme can be made even more efficient by using a global semantic network to determine the organization of peers in the graph topology. Namely, when peers describe their content in terms of this shared data-structure, peers are able to cluster themselves with similar peers. However, this approach based on a structured hypercube overlay has more maintenance overhead and is therefore also more sensitive to network dynamics than traditional broadcasting approaches.

[1] The Gnutella protocol specification v4.0. http://dss.clip2.com/GnutellaProtocol04.pdf/

Central registries A very popular approach is to have one central registry where systems can store their content descriptions or where the registry itself searches the network for the content descriptions. A well known example is Napster[2], which has one large repository which combines filenames with peers that offer those files for downloading. Such a repository can be seen as a kind of yellow pages, where each member in the network can look op the person or system that fulfills its needs. In small organizations, such an approach could work very well because the network is small and stable, so that the registry does not have to do much query processing and updates. In larger networks the approach is not very robust and has the same disadvantages as completely centralized approaches: undisclosed content, lack of privacy and censor possibilities.

Brokering The Multi-Agent community proposed the concept of 'broker agents' such as in InfoSleuth [Bayardo, Jr. et al., 1997]. These brokers semantically match information needs (specified in terms of some shared data-structure, e.g. an ontology) with currently available resources which are found by the broker or registered by the providing agents themselves. In InfoSleuth, agents advertise their services to the broker via the KQML [Finin et al., 1994] language. Broker agents respond to an agent's request for service with information about the other agents that have previously advertised relevant services. The literature on broker agents has a clear focus on finding services. Therefore, it is not surprising that the brokering approach is very popular in the literature on finding web-services which are semantically described [McIlraith et al., 2001]. An issue where the literature is not clear about is on how scalable and robust this approach is. In a network where millions of agents offer their services, one broker agent probably will not be enough and will have the same problems as with a central registry.

Super Peers/nodes An approach that looks very similar to brokering but with a different goal in mind, comes from the P2P research community. The technique, which seems to work well for file sharing, is to make use of the different capacities of the nodes in the P2P network. The fact, that some peers have more processing power, memory or network bandwidth than other peers is used to give them a harder job in the network. For example, KaZaa [Leibowitz et al., 2003] let peers voluntary be a super-peer where they have big routing tables (kind of yellow pages) where information is stored about the content of other peers. This approach is thus a hybrid solution between real P2P (all peers playing the same role) and centralized systems. Although better than broadcasting in a network without super-nodes, it remains broadcasting and therefore can be improved by techniques

[2]Napster. http://www.napster.com/about_us.html, 2002

that do more efficient routing described in the next paragraphs.

Distributed Hash Tables A technique from the P2P research community makes use of Distributed Hash Tables (DHT) [Ratnasamy et al., 2001, Rowstron and Druschel, 2001, Stoica et al., 2001, Aberer, 2001]. DHT's are based on the idea to route content (or a pointer to the content) to the peer whose identifier lies closest to the unique identifier of the content. This technique assumes that all peers share the same 'hash' function to assign a unique (mostly 128 bit) identifier to content, which could be anything like documents, music, URL's or words. The characteristic of this technique is that it allows to route content and queries in $\log(N)$ steps (where N is the number of peers in the network) to the right peers. A disadvantage of most DHT approaches is that they have high maintenance costs, due to continuous changes in the overlay network as a result of peers joining and leaving. P-Grid [Aberer, 2001] is a Peer-to-Peer search system based on a virtual distributed search tree, similarly structured as standard distributed hash tables however with an unstructured way of building the DHT overlay. Namely, they use randomized algorithms for constructing the access structure, updating the data and performing search. In this way probabilistic estimates can be given for the success of search requests, and search is more robust than the previously described DHT approaches against failures of nodes. A disadvantage of all DHT approaches is that content that is not hashed, cannot be found, which is especially a problem with full-text searching. Namely, in a document sharing case, one could roughly do three things (1) the file itself is hashed to a unique key with the disadvantage that the user has to know this key too, in order to find the file, or (2) the title of the document is hashed. This is also a problem because one typing error would result in a completely different hash key. (3) All the words in the document are hashed and the document or the location of the document is stored at the peers whose id's are closest to the hash keys of the words. Although now someone is able to find the documents that contain the keywords, the procedure of distributing the hash keys is not efficient because all these keys have to be distributed to the right peers in the network. Another disadvantage of DHT-based approaches is that load-balancing is not an emergent property of the topology. Due to the fact that content and queries follow a powerlaw distribution [Breslau et al., 1999], some peers (responsible for popular keys) are much more loaded than other peers that accidentally accidently are responsible for less popular keys. Imagine that one peer is responsible for the hashed key of 'Britney Spears', and that this peer joined the network with a slow 56K modem! Therefore, active load balancing policies have to be developed on top of DHT, [Byers et al., 2002] which is not needed for broadcast-based and expertise-based (described in next paragraph) alike approaches. Also, a pure DHT-based approach is less robust than broadcast-based and expertise-based approaches, because normally only one peer is responsible for one key, and

if that peer does not respond on queries (for example behind a firewall), no content can be found that is hashed to that key.

Semantic Overlay Networks Peers that keep pointers to other peers which have similar content as themselves form a Semantic Overlay Network (SON). Edutella [Nejdl et al., 2002] is a schema based network where peers describe their functionality (i.e. services) and share this with other peers. In this way, peers know about the capabilities of other peers and only route a query to those peers that are probably able to handle it. Although this provides complex query facilities, there is still no sophisticated means for semantic clustering of peers and their broadcasting does not scale well. Gridvine [Aberer et al., 2004] uses the semantic overlay for managing and mapping data and meta-data schemas, on top of a physical layer consisting of a structured Peer-to-Peer overlay network, namely P-Grid, for efficient routing of messages. In essence, the efficiency of the search algorithm is caused not by smart forwarding of the queries based on the semantic overlay, but by applying the underlying DHT approach for mapping terms to peers.

From here on, due to our focus, we only consider systems where the goal is an efficient search mechanism based on routing queries to peers that are semantically closest to the content of the query.

One way to do this is that the content of a peer is classified into a shared topic vector where each element in the vector contains the relevance for that given peer for the respective content. pSearch [Tang et al., 2002] is such an example where document indices are distributed through the P2P network based on document semantics generated by Latent Semantic Indexing (LSI) [Deerwester et al., 1990]. The search cost (in terms of different nodes searched and data transmitted) for a given query is thereby reduced, since the indices of semantically related documents are likely to be co-located in the network. In pSearch each peer has the responsibility for a range for each element in the semantic vector, e.g. $([0.2 - 0.4], [0.1 - 0.3])$. Now all vectors that fall in that range are routed to that peer, meaning that, following the example vector, the vector $[0.2333, 0.1939]$ would be routed to this peer and $[0.1322, 0.1939]$ not. One disadvantage of pSearch is that new documents in the network are 'folded' into the existing semantic vector, which means that when there are new terms in the documents that are not in the existing vector, they will not be used in the routing process. This means that when the content in the network changes frequently, also the computationally expensive LSI method has to be applied very often.

Another approach is based on random walk clustering, where peers with similar content are going to know each-other [Voulgaris et al., 2004]. The assumption is that queries posted by (the users of) peers are semantically closely related to the content of the peer itself. This results in a high prob-

ability that the neighbors of the peer (the peers in the cluster of that peer) have answers to the query. The problem of this approach in the domain of full-text searches, is what information a peer has to tell to another peer so that they are able to determine if they are related or not. When there is no shared data-structure (like a fixed set of terms) in which they can describe their content, the whole content has to be shared. This results in the fact that much data has to be shared between peers for determining closeness.

In contrast to the previous approach, the last SON approach that we discuss here lets peers describe their content in a shared set of terms. Mostly these terms are organized in a topic network or hierarchy making it possible to determine the semantic similarity between terms. Thus a peer has then a set of topics that describe its expertise. A peer knows about other peer's expertise topics by analyzing answers or advertisement messages [Haase et al., 2004c]. In this way peers form clusters of semantically related expertise descriptions. Given a query, a shared distance metric allows to forward queries (described by a shared set of terms) to neighbors of which their expertise description is semantically closely related to the query. The advantages of this approach are threefold:

- *Peer autonomy* Each peer can, in principle, have its own distance measure, peer selection mechanism and advertisement policy. This allows peers, for example to keep their neighbor list or similarity metric secret. Also peers can decide at any time to change their visibility on the network by sending advertisement messages. Also a peer can choose a shared distance matrix or create its own matrix that it could share with other peers. The quality of the routing process only depends on the overlap of the matrices, which means that the bigger the overlap, the better the routing performance.
- *Automatic load balancing* When some content is queried very often, most systems will also have copies of that content on their machines, due to downloading the results of the queries. This means that the semantic cluster on that content will contain many peers. In this way, load balancing is an emergent property of the network when popular content is distributed over many peers.
- *Robustness/fault tolerance* When peers leave the network or do not respond to a query, the only consequence is that they probably will not be asked a next time until they send new advertisement messages or are recommended by other peers. In contrast, most DHT approaches have to move routing tables to other peers in order to restore the overlay.

The problem of most SON approaches is that they either rely on a shared data-structure (distance matrix, topic-hierarchy or term-vector) in which the content has to be described and/or rely on the assumption that the queries of a peer are related to its own content and that a peer also has content that

allows it to cluster itself into the overlay. These assumptions are not always realistic. For example, imagine that a peer has some documents containing the word 'hound', but the shared data-structure only contains the term 'dog', then two things can be done: (1) extend the shared data-structure with the word 'hound' so that peers are able to query and describe their expertise with that term. (2) the functions that extracts the expertise description and abstract the queries should be intelligent enough to see that 'dog' is a good replacement for 'hound'. Note that in this case the original query still contains 'hound', but the routing mechanism uses the shared term 'dog' to route it to the peer that registered itself on that term. The problem with the first solution is that it will lead eventually to very large data-structures. To reduce this problem it is important to keep unimportant terms (e.g. stop-words) out the data-structure. The problem with the second solution is the strong dependency on the quality of the extraction and abstraction algorithms.

In the next section we introduce our SON approach where routing is also based on matching queries on content-descriptions described in terms that occur in a shared data-structure.

3.3 Expertise-Based Peer Selection based on a Shared Similarity Matrix

In this section we explain our expertise-based selection method by showing the individual building blocks of the system and describing the different advertisement and query policies.

3.3.1 Elements

Neighbor set Peers joining a network are assumed to know an initial random set of other peers called its initial neighbor set. More formally, each peer initially has a bootstrap neighbor list in a small, fixed-sized cache of entries (with typical value 5, 10, or 20). A cache entry contains the network address (i.e., IP-address and port) of another peer in the overlay. This set could, for example, be downloaded from a web-site, a gate-way peer or come with the download of the implementation. This set of peers is used by the advertisement policy that sends the expertise descriptions (discussed later).

Expertise Description Each peer has an expertise extraction function

$$\epsilon : docs_p \mapsto \mathcal{T}_p$$

which maps the content of the documents of the peer $docs_p$ into a set of terms. These terms describe the content of the peer's documents. This function could be based on an automatic NLP algorithm or be performed by a user that selects a set of terms which classifies the documents.To improve the semantic overlay, in our simulations we always let peers describe their expertise in terms occurring in a shared term similarity matrix on certain domain, which will be introduced in on of the next paragraphs. The assumption of a globally shared similarity matrix is clearly a limitation. We expect that our approach still works with only partial overlapping matrices, but we do not explore this in our experiments. So this is future work.

Query Abstraction For simplicity reasons we assume that a query posted by a user is just a set of words, thus without Boolean operators or other constructs except an implicit AND between the words. An answer on the query is therefore correct when all the words are in each of the result documents. The words in the query are mapped by a mapping function

$$\pi : \mathit{query} \mapsto \mathcal{T}_q$$

into a set of terms $\mathcal{T}_q \subset 2^{\mathtt{STRING}}$. These terms could exactly match the terms in the users query, but also be stemmed terms and with stop words removed. In our simulations we instantiate π by mapping queries into a set of terms occurring in a shared similarity matrix. This means that $\mathcal{T}_q \subseteq \mathcal{T}_{dom}$.

Term Similarity Matrix A term similarity matrix is a matrix $\mathcal{S}_{dom} : \mathcal{T}_{dom} \times \mathcal{T}_{dom}$ about a certain domain dom which contains semantic distances $[0, 1]$ between the set of (popular) terms $\mathcal{T}_{dom}$ in dom. In this way a peer is able to look up the semantic similarity $sim : (t_1, t_2)_{dom} \mapsto [0, 1]$ between two terms t_1 and t_2. These similarity values are used by the similarity function which will be described in the next paragraph. A user could download the domain matrices of interest (e.g. computer science or biology) from a web-site or develop its own although it is recommended to re-use similarity matrices as much as possible (will be discussed later). There are numerous ways to determine the semantic distance between terms. One way is to do it manually namely by letting a group of domain experts giving the distances. Another way is to use an ontology [Haase et al., 2004c] or another kind of semantic graph where the minimal path distance between topics is an indication of similarity. Our approach, which we will discuss later, is letting statistical algorithms automatically analyze a representative set of documents and determine the most descriptive terms and the similarity between them.

Similarity Function According to the expertise-based selection method, each peer has complete autonomy to choose its own similarity measure between two expertise descriptions or between an expertise description and a

query:

$$\sigma : (\mathcal{T}_e \bigcup \mathcal{T}_q, \mathcal{T}_e) \mapsto [0, 1]$$

where $\mathcal{T}_e$ consists of sets of terms from expertise descriptions and $\mathcal{T}_{eq}$ is the union of sets of expertise terms and abstracted query terms. In our simulations we assume that all peers in the network share documents about the same domain dom and therefore use the same term similarity matrix to determine the semantic similarity between terms. As said, future work should answer on what would be the effect of peers having different distance matrices from different domains and variations of them within the same domain. We instantiate σ in the following way:

$$\sigma(\bar{t}_{eq}, \bar{t}_e = \frac{1}{|\bar{t}_{eq}|} \sum_{t_i \in \bar{t}_{eq}} \max_{t_j \in \bar{t}_{eq}, \bar{t}_e} (sim(t_i t_j)_{dom}) \tag{3.1}$$

where $\bar{t}_e \in \mathcal{T}_e$ and $\bar{t}_{eq} \in \mathcal{T}_e \bigcup \mathcal{T}_q$. This formula takes the sum of the maxima of term similarities and divides it by the length of the expertise description $\bar{t}_e$. Note that this function is asymmetric, $\sigma(\bar{t}_{eq}, \bar{t}_e) \neq \sigma(\bar{t}_e, \bar{t}_{eq})$, because the elements in $\bar{t}_{eq}$ are all relevant in the match algorithm. The elements in $\bar{t}_e$ are only needed to determine how related $\bar{t}_{eq}$ is to them. For example, imagine a query [`ferrari, mercedes`] and an expertise description [`car, environment`]. The similarity matrix probably has values like: `(ferrari,car)=0.8, (mercedes,car)=0.9, (ferrari, environment)=0.3, (mercedes,environment)=0.3, (car,environment)=0.8, (ferrari,mercedes)=0.8`. In this case, $\sigma(\bar{t}_{eq}, \bar{t}_e) = \frac{1}{2} \times (0.8 + 0.9) = 0.85$. This value is high because for both the 'mercedes' and 'ferrari', the expertise of 'car' is relevant (the elements in the expertise descriptions are connected via a logical OR). If the query would be `car, environment` and the expertise description `ferrari, mercedes`, the value is $\frac{1}{2} \times (0.9 + 0.3) = 0.6$ which is much lower. This is reasonable because 'environment' is also part of the query (please recall that we assume that terms in the query are connected by a logical AND) and has only a low relevance to the terms 'ferrari' and 'mercedes' and therefore should lower the value.

3.3.2 Messages

There are three kinds of messages in our system needed to fulfill the functionalities of our algorithms:

- **Advertisement messages** These messages each contain a message identifier to prevent that a peer forwards the same message more than once, an expertise description, a list of peers that forwarded the message and the creator of the message. The purpose of making advertisements is that peers share expertise in order to increase the probability

that peers in the network know the right peers to forward a query to. This should reduce the number of (forwarded) messages needed to satisfy a query. We have to find the optimum between two sides: (1) There is no advertising at all, which results in the fact that peers only know about others expertise if they received an answer on a query. This often results in many forwarded query messages. (2) Peers initialize and forward advertisements to all peers they know, which results in the fact that peers have a rich overview of knowledge in the network and almost no query forwarding is needed, but many (forwarded) advertisement messages. In our experiments we try to find some indication on what is a good balance between the two, but we think that a good trade-off mainly depends on how static the content is in the network.

- **Query messages** These messages each contain a message identifier, a query, the query abstraction, the origin of the message and a list of peers that forwarded the message.
- **Query result messages** These messages each contain the identifier of the original query message, the creator of the message and the query result (or pointers to it).

3.3.3 Policies

In this subsection, we describe the policies that concern how queries and advertisements are handled in our pRoute system. These policies are needed to fulfill the basic requirements of our SON based P2P system: distribution and acceptance of expertise descriptions, distribution of queries and dealing with joining and leaving nodes. For each policy we give instantiations that we tested in our simulations. More precisely, for each policy we provide one instantiation based on a random selection method and one that uses the shared similarity matrix in the selection process. This allows us to see when and in which degree using the shared similarity matrix is beneficial. For all policies it holds that when a message is sent to a peer that is off-line, the sender will select a new peer from the neighbor list. This means that in a very dynamic system and/or a network where not many peers are on-line, a significant number of messages will be lost. We record this in our statistics and just count them as sent messages.

Query forwarding policy Besides that a peer tries to answer a query, peers also select peers to forward the query via the query forwarding policy. This policy is applied when a peer receives a forwarded query, or when a query is initialized by the peers user. We simulated the following instantiations:

- $QF1_{(h,n)}$: Queries are maximally forwarded h hops, each step to maximally n random neighbors. We call this respectively the forwarding

depth and forwarding *breadth*

- $QF2_{(h,n)}$: Queries are maximally forwarded h hops, each step to maximally n neighbors where peers whose expertise descriptions are semantically closest to the terms in the query.

Advertisement interval policy This policy regulates at which moment the peer starts to advertise its expertise. For all policies hold that when a peer (re)joins the network is advertises its expertise. More on this issue in the paragraph on the join/leave network policy.

- $AI1_{(r)}$: Advertising is periodically triggered after r expertise description updates.
- $AI2_{(s)}$: Advertising is only triggered when the similarity (see formula 3.1) between the expertise description from the last advertisement round and its new expertise description is lower than a certain threshold s (i.e. when the content of a peer has been significantly altered since the previous advertisement).

Advertisement distribution policy When the previous policy decides to advertise or when a peer receives an advertisement that it perhaps wants to forward, the advertisement distribution policy determines to whom to send the advertisements. This policy thus both applies to peers that want to advertise their own expertise and to peers that want to forward expertise descriptions to other peers.

- $AD1_{(h,n)}$: In the first hop, advertisements are sent to all neighbors and from the second hop, advertisements are distributed to a random subset of n neighbors until a maximum of h hops.
- $AD2_{(h,n,t)}$: In the first hop, advertisements are sent to those peers of which the expertise description has a similarity to the query above a threshold t. For two hops and more, the advertisements are sent to the n peers of which the expertise descriptions are semantically closest to the expertise description until a maximum of h hops.

Advertisement acceptance policy When a peer receives an advertisement it can decide to keep it or to ignore it. In the situation when the maximum number of n advertisements that a peer wants to store is not reached, we decide either to store every advertisement or the semantically close ones until all free advertisement slots are filled. After that, it can be the case that a new advertisement replaces an advertisement that exists in the receivers storage.

We tested the following two instantiations:

- $AA1_{(n)}$: Only those advertisements are stored of which the expertise descriptions are semantically closest to the receivers' own expertise description. This means that when the new advertisements description is closer than the worst description in the storage, it replaces this description.
- $AA2_{(n)}$: A received advertisement randomly replaces one of the stored advertisements.

Join/leave network policy In our experiments we only use one constant network join and leave policy:

- *Joining*: when a peer joins the network we assume that is has an initial (random) list of gateway-peers, or has the list of peers that it had before it left the network. When the peer successfully joined the network it starts the advertising initialization policy.
- *Leaving*: when a peer leaves the network, by a network failure or intensionally initialized by the user, we do not assume any policy: the peer is just unreachable. We implemented the policy that when a peer sends a message to an off-line peer, it will recognize this and remove the peer from its neighbor list.

Now that we have introduced all elements of the pRoute system, we show a method to build a term similarity matrix which is one of these elements.

3.4 Method for making a Term Similarity Matrix

This section describes a method for automatically building a term similarity matrix.

Literal matching schemes suffer from synonyms and noise in documents. Latent Semantic Indexing (LSI) overcomes these problems by using statistically derived concepts instead of terms for retrieval. LSI uses Singular Value Decomposition (SVD) [Deerwester et al., 1990] to transform a high-dimensional document vector into a lower-dimensional semantic vector, by projecting the former into a semantic subspace. In this section we adopt a method using SVD to make a term similarity matrix on a certain domain which can be shared among a group of peers. This method also is used to calculate the shared matrix for our experiments.

Our method contains five steps:

1. **Get representative document set.** For text collections spanning many contexts (e.g., an encyclopedia), the number of terms is often much

greater than the number of documents: $t \gg d$. However, in our case where the Internet is the document collection, the situation is very likely to be reversed: there are much more documents than terms. Even if we divide the set of documents in popular domains, it is likely that there are much more documents than items. In that case we only need a representative subset of the documents to get most terms used in that domain.

2. **Extract the keywords for each document.** Use a keyword extraction algorithm like those which are developed in the Natural Language Processing (NLP) research domain [Cunningham et al., 1996], to extract for each document a set of keywords which are the terms that describe the content of the documents. The size of the set depends on the trade-off between the size of the term-by-document matrix and the possibility to classify the assumed queries into one or more of the terms. Many keywords will lead to a big term-to-document matrix, for which it becomes computationally very expensive to calculate the term-to-term similarity matrix. However if there are many terms, the algorithm of abstracting a query into these terms will be very easy because many of the strings in the query (after stemming) match the terms in the matrix. Based on our experiences with the SVD tool 'SVDLIBC'[3], we can say that for 100.000 documents containing together 200.000 terms it is still doable to do the calculations as described in step four of our method (AMD dual Athlon 3000+ 2G RAM roughly taking 1 hour)

3. **Create a term-by-document matrix.** Create a database containing the total d document sets described by t terms (the union of all document keyword sets) is represented as a $t \times d$ *term-by-document matrix* $\mathcal{A}$. The d vectors representing the d document descriptions form the columns of the matrix. Thus, the matrix element a_{ij} is the weighted frequency at which term i occurs in document description j.

4. **Calculate the SVD of the term-by-document matrix.** Each document vector in the term-by-document matrix can be placed as a point in a large multi-dimensional *term space*. The Singular value decomposition (SVD) algorithm can be used to reduce the term space to a smaller number of dimensions. In doing so, terms that are semantically similar will get squeezed together, and will no longer be completely distinct. We re-use the description of [Tang et al., 2002] to describe the SVD algorithm. Let $\mathcal{A}$ be a $t \times d$ matrix as described in the previous step, whose entry a_{ij} indicates the importance of term i in document j. Suppose the rank of $\mathcal{A}$ is r. SVD decomposes $\mathcal{A}$ into the product of three matrices, $\mathcal{A} = \mathcal{U}\Sigma\mathcal{V}^T$, where $U = (u_1, ..., u_r)$ is a $t \times r$ matrix, $\Sigma = diag(\alpha_1, ..., \alpha_r)$ is an $r \times r$ diagonal matrix, and $\mathcal{V} = (v_1, ..., v_r)$ is a $d \times r$ matrix. α_i's are $\mathcal{A}$'s singular values, $\alpha_1 \geq \alpha_2 \geq ... \geq \alpha_r$.

[3] SVDLIBC: http://tedlab.mit.edu/ dr/SVDLIBC/ version 1.34 (2004)

Now the 'trick' of LSI is to reduce the dimensionality of the matrix by simply removing the singular values that have a low value. The result of this reduction is term clustering based on similarity, which we discuss in the next step. In [Berry et al., 1999] we can find a good overview how such fundamental mathematical concepts from linear algebra can be used to manage and index large text collections.

5. **Use the SVD to generate the term-by-term similarity matrix** Before we explain how the SVD can be used for generating a term-by-term similarity matrix from $\mathcal{A}$, we first explain how it already can be done by simply comparing the term vectors in $\mathcal{A}$. The term-to-term comparison is carried out by computing the cosines of the angles ω_{ij} between all pairs of term vectors i and j:

$$\cos\omega_{ij} = \frac{(e_i^T\mathcal{A})(\mathcal{A}^T e_j)}{\|\mathcal{A}^T e_i\|\|\mathcal{A}^T e_j\|}$$

where i and j is the number of terms in $\mathcal{A}$ and where e_l denotes the lth canonical vector of dimension t (the lth column of the $t \times t$ identity matrix). The cosines can now be listed in a $t \times t$ symmetric identity matrix $\mathcal{S}$, where $\mathcal{S}_{ij} = \cos\omega_{ij}$. The entry $\mathcal{S}_{ij}$ reveals how closely term i is associated with term j. If the entry is near 1, the term vectors for the two terms are nearly parallel meaning that the terms are closely correlated. The problem with this approach is that in our case $\mathcal{A}$ is sparse because on average each document description only contains a small subset of the terms. This means that many entries in $\mathcal{S}$ will be 0. Recall that our goal of the similarity matrix is to find terms that are related to other terms, which means that it would be better that for many terms a set of terms can be found which are at least not 0 but still reflect a correct similarity value. This would allow the advertisement- and query forwarding mechanisms to route messages to peers on which the expertise descriptions are close to the content of the received advertisement or query. This is where SVD becomes a good candidate, because reducing the rank of the matrix uncovers the associations among terms in a large collection of terms [Deerwester et al., 1990]. Because it falls beyond the scope of this paper to go into detail on LSI, we only show how this term matrix based on the reduced rank SVD is constructed. By the reduced-rank method, $\mathcal{A}_k = \mathcal{U}_k\Sigma_k\mathcal{V}_k^T$ replaces the original term-by-document matrix $\mathcal{A}$, where k is the reduced number of singular values (and therefore the rank of the new matrix $\mathcal{A}_k$). Recall that the columns of $\mathcal{U}_k$ forms a basis for the column space of $\mathcal{A}_k$, and so those columns could be used in place of the columns of $\mathcal{A}_k$ for query matching. In the same way, the rows of $\mathcal{V}_k$ are a basis for the row space of $\mathcal{A}_k$ and so can replace the rows of $\mathcal{A}_k$. Thus, in a reduced-rank approximation, the term-by-term can be constructed by calculating for each pair i, j the cosine [Berry et al., 1999] :

$$\cos w_{ij} = \frac{(e_i^T \mathcal{U}_k \Sigma_k \mathcal{V}_k^T)(e_j^T \mathcal{U}_k \Sigma_k \mathcal{V}_k^T)}{\|\mathcal{V}_k \Sigma_k \mathcal{U}_k^T e_i\|_2 \|\mathcal{V}_k \Sigma_k \mathcal{U}_k^T e_j\|_2} = \frac{(e_i^T \mathcal{U}_k \Sigma_k)(\Sigma_k \mathcal{U}_k^T e_j)}{\|\Sigma_k \mathcal{U}_k^T e_i\|_2 \|\Sigma_k \mathcal{U}_k^T e_j\|_2}$$

for $i = 1, ..., t$ and $j = 1, ..., d$. Defining $b_j = \Sigma_k \mathcal{U}_k^T e_j$, we have

$$\cos w_{ij} = \frac{b_i^T b_j}{\|b_i\|_2 \|b_j\|_2}$$

for $i = 1, ..., t$ and $j = 1, ..., d$.

In this section we proposed a method to automatically build a term similarity matrix, which is an important element in the pRoute system. In the next section we will show our experimental setup and introduce our simulation platform by which we test the characteristics of our approach.

3.5 Experimental Setup

We designed and implemented a simulation platform to test the performance of the different policies. As we will see, the semantics-based policies always outperform their random-based counterparts, although it will turn out to be situation dependent which combination of policies is the optimal one. For example, when the content of peers in the network remains fairly static, it is worthwhile to 'invest' in the visibility of peers by sending relatively many advertisement messages resulting in a reduction of the number of query forwards due to better peer selection. Also the recall rate that a user likes to see will turn out to be an important factor in selecting the policy.

3.5.1 Scenario and the data-set

Although the characteristics of our system are probably independent from the domain, we base our experiments on a realistic scenario where we can also easily get the needed data-set. Namely, we simulate a scenario in which our P2P system runs in a scientific domain where researchers share their publications with other researchers via a P2P network. This means that in our simulation researchers are represented by peers that share the publications of those researchers and publications that are interesting for them. We do this by letting the peers also include a (small) subset of the documents that are returned for queries posted by those peers. Each time when a peer adds a document of its user or includes a set of documents by the query process, the expertise description of that peer will be updated. The expertise

descriptions will contain the most important terms of the researcher it represents, which reflects the content of the documents it shares. In this way we get a realistic overlap of document ownership and also a realistic growth of the document set over time. The queries that peers post are based on the content of those peers which supports the assumption that in our research scenario, the interests and expertise of users are closely related. We make the queries by selecting some terms which are in the document descriptions of that peer.
Our data set contains a crawled set of research articles from the computer-science domain sorted per author. The crawling procedure was first to extract the authors and titles from the DBLP resource[4], a popular database containing bibliographic items, and use Citeseer[5] to fetch the PDF document. We used the Unix tool PDF2Text to extract the textual version of the PDF documents. Only when the tool was able to extract the text, the entry (author + text) is added to our dataset.

Peer set, document description set The document set contains 101,133 scientific documents in the computer science domain, namely a subset of those cited in the DBLP-database. The document crawl procedure is based on a breadth-first search over co-authors, starting by one author. More precisely, we started by crawling and storing all documents and co-authors from the author 'Maarten van Steen'. Next, for all the co-authors we crawled their documents and their co-authors and removed the doubles. This procedure is continued until a given maximum that we put on 82.054 peers. The characteristics of our data-set are as follows: Each document in our data-set has on average 2.71 authors (see Figure 3.3), and each author has on average 3.22 documents (see Figure 3.4). We used an NLP tool called 'TextToOnto' [Maedche and Staab, 2001] to extract for each document a set of descriptive terms (around 10 terms per document, see Figure 3.2), which resulted in 175,008 unique terms.

Shared similarity matrix In our experiments we assume that peers all share the same term similarity matrix created via the methodology of the previous section. We used a random subset of the 101K document descriptions from the previous paragraph as a source for creating this similarity matrix. The number of documents that is selected clearly needs to cover most of the shared terms from the 101K documents. We define a term as a *shared term*, when it is shared by at least three peers (an arbitrary number). This resulted in a set of 25K shared terms (see Figure 3.1). Now that we have this shared term set, we have to find out how much we can reduce the document set before the number of shared terms decreases below an acceptable ratio. Also here we choose an arbitrary threshold of 90%, which means

[4]DBLP: http://dblp.uni-trier.de/
[5]CiteSeer: http://citeseer.ist.psu.edu/

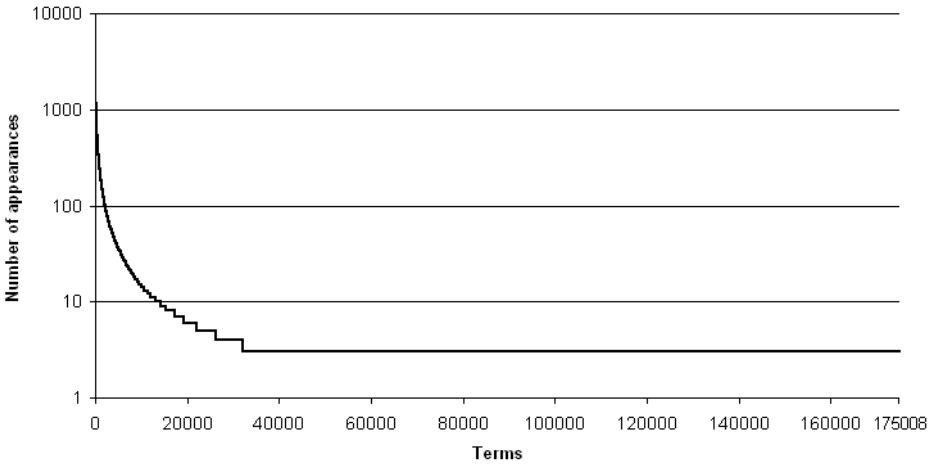

Figure 3.1: **Term popularity.** This figure shows the distribution of the number of documents per term. There are a few terms that occur in many term vectors, and many terms only occur in three term vectors. Most terms (175008-32718=142290) have three matching vectors (we fixed this minimum to this value and discarded the terms that have less matching vectors).

that we stop reducing the set when the reduced shared term set covers less than 90% of the original shared term set. In our case we could randomly reduce the document set by 90% which means that in our case we only needed 10K documents to get 92% of the shared terms. This reflects the realistic situation that the initial set of shared terms is created based on a small initial data-set. Now for each of the 10K documents a term vector is created of 25K elements, with value 1 at places where the term occurs in the document and value 0 (most of them) where the term does not occur. We use the method as described in the previous chapter to calculate the Term $\times$ Term matrix, based on these 10K document vectors.

Expertise descriptions As said in previous sections, an expertise description describes the 'expertise' of a peer by a set of terms from the shared distance matrix. The expertise descriptions in our experiments are made by taking the union of all document descriptions of the peers and filter out the terms which are not in the distance matrix.

Query abstractions We create the set of query abstractions for a peer p by taking random sets of 3-5 terms from randomly chosen document descriptions d_{desc} owned by the peer, where the terms should also occur in the term matrix.

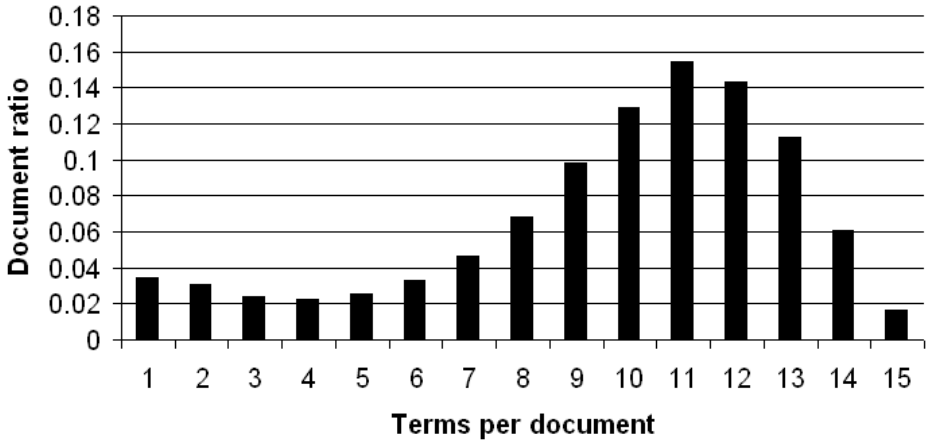

Figure 3.2: **Terms per document.** This figure shows the distribution of the number of terms per document(vector). Most documents have around 9-13 terms in its vector. We fixed the maximum number of terms to 15.

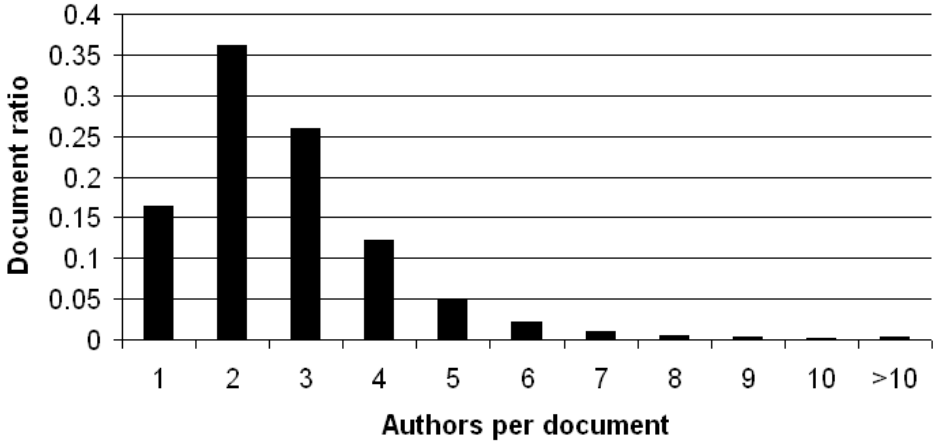

Figure 3.3: **Authors per document.** This figure shows the distribution of the number of authors per document. Most documents have 1-4 authors.

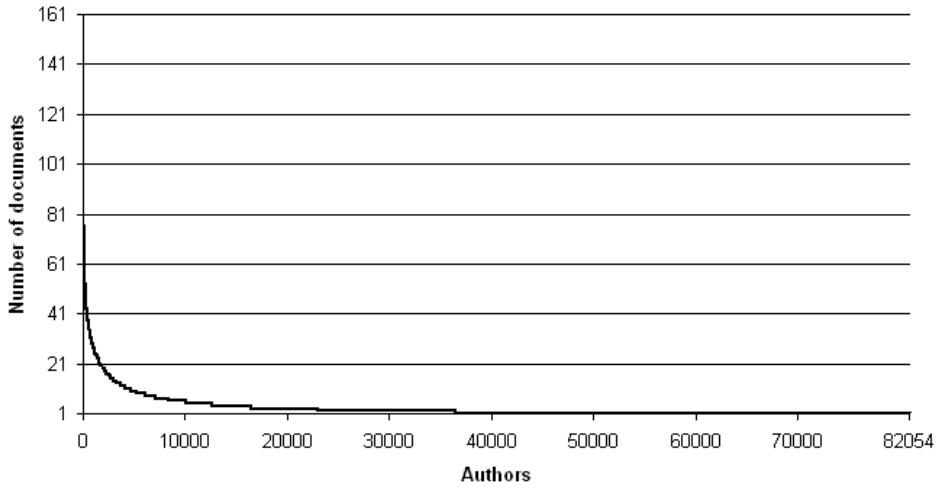

Figure 3.4: **Documents per author.** This figure shows the distribution of the number of documents per author. There are only a few authors that have more than 10 documents (in our data-set).

We believe that the creation of this data-set in itself is a contribution of this paper. The data-set is available from the authors on request.

3.5.2 Simulation platform and Scenario Generator

We wrote a simulation platform to examine the different characteristics of our pRoute system. The platform is composed out of two elements: the 'scenario generator' and the 'scenario executor'. The generator is used to create a sequence of events to be used as input for the scenario executor. Having a different program to create the scenario yields several benefits compared to having a single program for both creating a scenario and applying it as input to our scenario executor:

- Firstly, comparing the output of the Peer-to-Peer system is more straightforward when exactly the same sequence of external events is used as input.
- Secondly, it yields performance gains when simulating several configurations of the Peer-to-Peer system.
- Finally, its output, the scenario, is in XML format, making the platform reusable and allowing user modifications to the scenario.

Each scenario script contains a large set of events from three different types, query events, document inclusion events and 'leave/join network' events, which normally would all be initiated by the users in a real network. A document inclusion event lets a peer include one of its own documents (i.e. of which the user is the author of that document) into its existing set of documents that it already shares on the network. A query event has the effect that a peer sends a query on the network which has to be logged for our statistics. But also it means that when documents are returned as an answer on the query, some of them will be included into the shared document set. A 'leave/join network' event has the effect that a peer temporally disconnects from the network. A system parameter in the scenario executor determines the time that the peer stays off-line. This means that after a certain amount of time, the peer goes back on-line. The scenario generator is described by algorithm 1, which we briefly describe as follows: In step 1, a set of document descriptions is generated from the complete set of documents that possibly will be shared in the network. These descriptions are sets of terms that occur in the shared term matrix. Step 2 combines authors (represented as peer identifiers) with their documents. Step 3 initializes an empty set of scenario events. From step 4 to step 27, the set is iteratively filled with three different types of events until a maximum number of events (which is a system parameter, given in step 4) is reached or when the set of documents that a peer may add to the network is empty. The type of event that is added per iteration is determined by the $determineEventType()$ function (step 8), which is a probabilistic function that decides if the next event that will be added to the script is a query event, a document inclusion event or a leave/join network event.

The scenario executor parses a scenario script which is generated by the scenario generator. Mechanisms that implement various policies can be easily added without modification to existing code. The simulator runs in a single process. Furthermore, it is capable of handling up to 100,000 Peers and hundreds of thousands of documents. Performance has been an important issue on the development of the simulator. We do not assume a reliable message delivery, which in our case is limited to the situations where peers send messages to peers that are not online. In such a case, besides that the message is counted by the statistics, the sender also knows that the message was not received. In that case the peer, to which the message was sent, is removed from the neighbor list. We assume the following simplified ordering of messages: a message only can be forwarded for the n_{th} time if all previous messages have been forwarded for the $(n-1)_{st}$ time. This assumption is made because it simplifies our simulator and considerably improves its execution time. We do not expect any significant changes in the results when the message delivery is more parallel.'

Algorithm 1 Scenario script generator

1: Let $D_{desc} := describe(D, S)$ be the set of document descriptions generated from the complete set documents D by terms occurring in the shared term similarity matrix S
2: Let $C := D_{desc} \times ID$ the set of tuples that combines each document description $d \in D_{desc}$ with the peers (identified by peer identifiers $p \in ID$) that have a document that matches the description
3: Let $E := \emptyset$ be a list of scenario events
4: Let $TotalNrOfEvents$ be the total number of events that the generated scenario script will contain.
5: **while** $E.size < TotalNrOfEvents$ AND $C \neq \emptyset$ **do**
6: `//select a random tuple from` C
7: $[p, d] := getRandomTuple(C)$
8: $eventType := determineEventType()$
9: **if** $eventType == events.query\_event$ **then**
10: `//get a random set of terms` q `from` d
11: $q := getRandomSubset(d)$
12: `//create a query event for peer` p `and query` q E
13: $queryEvent := createQueryEvent(p, q)$
14: `//add the query event to` E
15: $E := E.add(queryEvent)$
16: **if** $eventType == events.inclusion\_event$ **then**
17: `//create an inclusion event for peer` p `and document` d
18: $inclusionEvent := createInclusionEvent(p, d)$
19: `//add the doc inclusion event to` E
20: $E := E.add(inclusionEvent)$
21: `//remove the tuple from the` C
22: $C := C \setminus [p, d]$
23: **if** $eventType == events.leave\_join\_network\_event$ **then**
24: `//create a leave/join network event for peer` p
25: $leaveJoinEvent := createLeaveJoinEvent(p)$
26: `//add the leave/join network event to` E
27: $E := E.add(leaveEvent)$
28: return E

3.5.3 Evaluation criteria

We evaluate the different policies and the influence of other parameters by indicating user satisfaction (measured as document description recall), system efficiency and robustness. Please recall that each document and query is described by a set of terms from the shared term similarity matrix. Our precision and recall measures are based on these descriptions and not on the actual queries and documents. We expect that this is not a problem because the most important stemmed terms are in the 160K terms from the similarity matrix and many queries will contain (after stemming and removing the stop-words) these terms. Our evaluation is reflected by the following criteria:

- **Document description precision**

  $$Precision_{Doc} = \frac{|D_{relevant} \bigcap D_{returned}|}{|D_{returned}|}$$

  indicates how many of the returned document descriptions are relevant, with $D_{relevant}$ being the total set of matching document descriptions in the network and $D_{returned}$ being the set of returned document descriptions for queries. A document description is seen as relevant when all the terms in the query description occur in the document description. We determine the set of relevant documents $D_{relevant}$ by evaluating the query against a centralized database which contains the complete set of document descriptions. In our model we work with exact queries, therefore only correctly matched documents are returned. The precision will therefore always be one, since $D_{returned} \subseteq D_{relevant}$:

  $$Precision_{Doc} = \frac{|D_{returned}|}{|D_{returned}|} = 1$$

  and will therefore not be an evaluation criterion.

- **Document description recall** Doc_{rec}

  $$\frac{|P_{returned} \bigcap P_{relevant}|}{|P_{relevant}|} = \frac{|P_{returned}|}{|P_{relevant}|}$$

  The recall on the document level states how many ($|P_{returned}|$)of the total amount of relevant documents for the queries ($|P_{relevant}|$) are returned and can be seen as an indication for user satisfaction. As we will see, the optimal policies differ for distinct recall levels. Therefore, recall level can also be seen as a requirement.

- **Peer precision** $Peer_{prec}$

  $$\frac{|P_{relevant} \bigcap P_{queried}|}{|P_{queried}|} = \frac{|P_{relevant}|}{|P_{queried}|}$$

The peer precision states the percentage of queried peers during the query process which are 'correctly' queried , i.e. where ate least one document of the peer was correctly matched. This measure can be seen as an indication for the efficiency of the search process.

- **Average number of messages per query** Q_{msg} The average number of messages used to resolve a query. It is an indication of the efficiency of the system.
- **Average number of advertisement messages per peer per experiment** A_{msg} The advertisement initialization policy decides if the peer should (re)advertise its expertise. Normally the advertisement procedure starts when the expertise description of a peer changes, for example caused by a significant change of content that a peer stores or when the peer goes back online. When a peer decides to advertise its expertise, its advertisement distribution policy decides to which peers the advertisements should be send to. The average number of advertisement messages per peer during one simulation experiment is based on the multiplication of the total number of advertisement initializations and the number of messages per initialization. This number can be seen as the construction costs and maintenance costs of the semantic overlay for that experiment.

3.5.4 A short note on the simulation method

When we finished our simulation platform and started different combinations of policies to get an idea of the behavior, we discovered that the results of the experiments are not only determined by the individual policies but also strongly depends on the combination of different policies. For example, an advertisement forward policy $A1$ combined with a query forward policy $Q1$ could give very good results like also advertisement forward policy $A2$ combined with query forward policy $Q2$, but the combination of $A1$ and $Q2$ could give very bad results. This means that the way to combine policies is as important as the choice of the individual policies themselves. Due to the fact that simulating all different possibilities together with all different parameters for each individual policy would lead to an unacceptable amount (millions) of experiments, we choose an iterative local search selection procedure. Namely, first we generated a pool of hundreds of experiments instantiated with some possible good combinations of policies and their parameters. These choices are made based on intuition and with the goal in mind of showing the influence of using semantics in the whole process. After that, we generated a new pool of experiments, by taking the best performing combinations of the previous step and make some changes in their settings to see how dependent the result was on the individual parameters. In this way we cannot guarantee that we found the best combination of policies together with their optimal settings, however at least can show a variety of

description	default value
total number of peers in the system	27,351
average peer availability (α)	40%
average nr of reconnects per peer (β)	75

Table 3.1: Default parameters in experiments

experiments showing the difference between policies using semantics in the advertisement and querying process and those base on random behavior. In Table 4.2 the default settings are shown for the different experiments. Sometimes we deviate from these settings, and if so, we mention these alternated settings.

Now that we have shown how our simulation platform, data-set and evaluation criteria look like, we can present the results of our experiments in the next section.

3.6 Results

Random-based strategies Table 3.2 shows how the system performs when it uses random query- and advertisement strategies and can be seen as our baseline settings.

- Advertisements are needed to make peers visible in the network. We only tested the settings where the number of advertisement hops are two or three. We skipped the one-hop experiments because we can already know that these will perform bad. Namely, each peer knows in the beginning a set of random pointers (peers) to which it can send its advertisements. If the receiving peer does not forward the advertisements, the topology always will be the initial topology which gets outdated after a while. Which means that a new set of pointers has to be found (e.g. downloaded from a web-site) somewhere, which is an undesirable option due to the centralized characteristics of it. When we compare the recall from experiments 1,2,3 with experiments 4,5,6 (where we increase the advertising depth), we can see a 20% drop in the recall. This is because the visibility of the peers that send the latest advertisements becomes too strong because their advertisements will be sent to a an unacceptable large number of peers. This results in less visibility of other peers that sent previously advertisements. For example in experiment 6, there are 1092 messages sent per advertisement initialization. Given that on average there are 20000 peers online, the advertisement storages of the individual peers will be updated too often. Therefore, only the last advertisements sent in the network will

be stored, resulting in a smaller fraction of the peers being known. Summarizing: be careful with the number of advertisement messages. When there are too much forwards of the same advertisement, not only will the network traffic be increased, it will also result in too many existing advertisements being replaced by it, thus making peers either too visible or too little visible.

- A second observation is that, besides the recall, the precision increases when there are more query hops. This is because when there are more query hops, more results will be returned. Due to the fact that in our simulation some of the query results are added to the document of the querying peer, it will lead to more copies of the documents which automatically increases the chance that a peer contains an answer for a query. We deliberately keep this inclusion rate small to keep this effect into reasonable boundaries.

Random-based query routing combined with semantic-based advertisements Table 3.3 shows the results of combining the random-based query routing strategy with the semantic-based advertisement strategies.

- The results of experiment 7 to 11 show that replacing random strategies by their semantic strategies increases the performance, however the gain differs per choice. For example, it seems that it is better to choose a semantic advertisement acceptance policy instead of a semantic advertisement distribution policy (compare experiment 8 with 9). Choosing both seems to be the best solution (experiment 10). Experiment 11, compared with experiment 10, shows that doing semantic advertisement initialization reduces the number of advertisements but also the recall and precision. These results indicate that replacing random policies by their semantic counterparts has a positive effect.
- Experiment 11, 12 and 13 show the influence of different query hops on the number of query messages and the recall. They indicate that in order to double the recall, the number of query messages need to be more than tripled.
- Experiment 11, 14 and 15 indicate that there is a positive effect of the number of advertisements on the recall and precision. This confirms the positive effect of clustering peers with the same expertise: the recall and precision of this clustering is higher because in our system peers normally ask questions related to their expertise and therefore can benefit from the fact that neighbors have already a good chance to answer the query.
- Experiment 16 to 22 show the influence of the number of neighbors that a peer knows. The results indicate that this factor is important for

Nr	Simulation setting				$Peer_{prec}$	Doc_{rec}	Q_{msg}	A_{msg}
1	$QF1_{(\underline{3},3)}$	$AI1_{(1)}$	$AD1_{(2,5)}$	$AA1_{(80)}$	0.0076	0.0093	44	214
2	$QF1_{(\underline{4},3)}$	$AI1_{(1)}$	$AD1_{(2,5)}$	$AA1_{(80)}$	0.0082	0.0326	133	211
3	$QF1_{(\underline{5},3)}$	$AI1_{(1)}$	$AD1_{(2,5)}$	$AA1_{(80)}$	0.0085	0.1013	384	214
4	$QF1_{(\underline{3},3)}$	$AI1_{(1)}$	$AD1_{(3,5)}$	$AA1_{(80)}$	0.0070	0.0086	42	1121
5	$QF1_{(\underline{4},3)}$	$AI1_{(1)}$	$AD1_{(3,5)}$	$AA1_{(80)}$	0.0072	0.0281	124	1099
6	$QF1_{(\underline{5},3)}$	$AI1_{(1)}$	$AD1_{(3,5)}$	$AA1_{(80)}$	0.0078	0.0835	357	1092

Table 3.2: **Random query routing with random advertising.** For the meaning of the simulation settings, please look at Subsection 3.3.3

the performance of the system. Namely a too small number of neighbors makes that a peer cannot remember a sufficient set of the relevant peers. However the number can also be too large. Namely this makes that a peer stores relatively too many irrelevant peer advertisements, which increases the chance that, due to the random query forwarding process, peers are selected that have no expertise on the query (experiment 22).

Semantic query routing combined with random-based advertisement strategies Table 3.4 shows the results of combining the semantic-based routing strategy with the random-based advertisement strategy. The results indicate that this combination outperforms the previous two combinations. Due to this random-based advertising, there is no clustering of expertise, meaning that each peer only knows a random set of peers where a new advertisement replaces randomly an old advertisement. The drop in recall for experiments with a high number of hops (compare experiment 26 with 27) is likely to have the same reason as given at the experiments that combines random advertisement and query strategies. Namely, peers that recently advertised are unacceptably oppressing the other peers that have advertised before due to the enormous amount of copies of the same advertisement caused by the large number of forwards. As before, this shows that a careful balancing of parameters is required.

Semantic query routing, semantic advertisements Table 3.5 shows the results of combining semantic query policies with semantic advertisement policies. The results indicate that this combination outperforms all the previous three combinations.

- Experiments 28 to 33 show the effect of changing the random policies from the advertisement process by their semantic counterparts. Result 33 indicates that by replacing all of the random policies by semantic ones, a good recall can be achieved (42%) together with a low number of advertisement and query messages.
- Experiments 33 and 34 show that changing the similarity threshold in the advertisement initialization policy increases the recall but also the number of advertisement messages.
- Experiments 35 and 36 show the effect of playing with the number of query hops. The difference between two and three hops (experiment 32) results in this case in a recall increase from 27% to 42%. By doubling the number of messages again by increasing the number of hops from three to four (experiment 36), the recall increases to almost 50%.
- The effect of increasing the number of advertisements is shown in result 37 and 38. Although there is a positive effect (compare result 37

Nr	Simulation setting				$Peer_{prec}$	Doc_{rec}	Q_{msg}	A_{msg}
7	$QF1_{(4,3)}$	$AI2_{(0.8)}$	$AD1_{(3,5)}$	$AA1_{(15)}$	0.0059	0.0177	125	969
8	$QF1_{(4,3)}$	$AI1_{(1)}$	$AD2_{(3,5,0.2)}$	$AA1_{(15)}$	0.0054	0.0130	142	412
9	$QF1_{(4,3)}$	$AI1_{(1)}$	$AD1_{(3,5)}$	$AA2_{(15)}$	0.0245	0.1103	123	1021
10	$QF1_{(4,3)}$	$AI1_{(1)}$	$AD2_{(3,5,0.2)}$	$AA2_{(15)}$	0.0181	0.0809	136	464
11	$QF1_{(4,3)}$	$AI2_{(0.8)}$	$AD2_{(3,5,0.2)}$	$AA2_{(15)}$	0.0148	0.0647	137	422
12	$QF1_{(3,3)}$	$AI2_{(0.8)}$	$AD2_{(3,5,0.2)}$	$AA2_{(15)}$	0.0147	0.0309	47	423
13	$QF1_{(5,3)}$	$AI2_{(0.8)}$	$AD2_{(3,5,0.2)}$	$AA2_{(15)}$	0.0130	0.1299	391	403
14	$QF1_{(4,3)}$	$AI2_{(0.8)}$	$AD2_{(2,5,0.2)}$	$AA2_{(15)}$	0.0073	0.0192	126	55
15	$QF1_{(4,3)}$	$AI2_{(0.8)}$	$AD2_{(4,5,0.2)}$	$AA2_{(15)}$	0.0188	0.0904	132	807
16	$QF1_{(4,3)}$	$AI2_{(0.8)}$	$AD1_{(3,5)}$	$AA2_{(4)}$	0.0119	0.0249	51	41
17	$QF1_{(4,3)}$	$AI2_{(0.8)}$	$AD1_{(3,5)}$	$AA2_{(6)}$	0.0188	0.0580	73	87
18	$QF1_{(4,3)}$	$AI2_{(0.8)}$	$AD1_{(3,5)}$	$AA2_{(8)}$	0.0232	0.0795	86	121
19	$QF1_{(4,3)}$	$AI2_{(0.8)}$	$AD1_{(3,5)}$	$AA2_{(22)}$	0.0240	0.1137	111	309
20	$QF1_{(4,3)}$	$AI2_{(0.8)}$	$AD1_{(3,5)}$	$AA2_{(30)}$	0.0250	0.1257	115	413
21	$QF1_{(4,3)}$	$AI2_{(0.8)}$	$AD1_{(3,5)}$	$AA2_{(60)}$	0.0158	0.0887	118	577
22	$QF1_{(4,3)}$	$AI2_{(0.8)}$	$AD1_{(3,5)}$	$AA2_{(80)}$	0.0085	0.0395	139	373

Table 3.3: Random query routing with semantic-based advertising policies

Nr	Simulation setting				$Peer_{prec}$	Doc_{rec}	Q_{msg}	A_{msg}
23	$QF2_{(\underline{3,3})}$	$AI1_{(1)}$	$AD1_{(3,5)}$	$AA1_{(80)}$	0.0669	0.1213	40	1104
24	$QF2_{(\underline{4,3})}$	$AI1_{(1)}$	$AD1_{(3,5)}$	$AA1_{(80)}$	0.0668	0.2299	105	1104
25	$QF2_{(\underline{5,3})}$	$AI1_{(1)}$	$AD1_{(3,5)}$	$AA1_{(80)}$	0.0674	0.3454	211	1084
26	$QF2_{(4,3)}$	$AI1_{(1)}$	$AD1_{(\underline{2,5})}$	$AA1_{(80)}$	0.0637	0.2724	113	213
27	$QF2_{(4,3)}$	$AI1_{(1)}$	$AD1_{(4,5)}$	$AA1_{(80)}$	0.0665	0.1981	98	3486

Table 3.4: Semantic query routing with random-based advertising policies

with 32), it seems that the topology gets saturated at a certain point because the difference in recall between result 32 and 38 is almost the same although the number of advertisements almost doubled.

Influence of the number of neighbors Table 3.6 shows the effect of varying the number of expertise descriptions (also called the number of neighbors) that a peer can store. As it can be seen, an increase improves the recall and precision, without an increase in the number of query messages although with an increase of the number of advertisements. This increase is because the number of neighbors selected at the first advertisement hop depends on the number of neighbors in the storage: the more neighbors, the bigger the chance that peers are above the similarity threshold. The trade-off between recall and the number of messages is off course dependent on the requirements of the user of the system. However, the results indicate that the positive effect on the recall by sending more advertisement messages in combination with a larger neighbor storage starts to diminish after 60 neighbors (compare experiment 42 with 43). Probably this result is dependent on the number of peers in the network, where the trade-off in larger networks lies at a higher number of peers.

Influence of the network size Table 3.7 shows the results for varying the network size (in the other tables, the network size is fixed at 27K peers, see Table 4.2). The results indicate the precision almost stays the same for different network sizes. That the recall drops a bit is because more peers means more content, which means that more peers needs to be visited to get the relevant results. In other words, given that the maximum number of query forwards stays the same during all the shown experiments, it can be derived that the recall should decrease. Experiment 49 shows a large decrease in recall and advertisement messages compared to experiment 44 to 48 which has the following cause: When there are not enough advertisement hops, the relevant experts for the advertiser (i.e. the peers that have similar expertise as the sender) are not reached at an advertisement phase. This results in the situation that both sides will not know each-other (the advertising peer and the related experts). This is because when the peers that are reached at the advertisement phase are not relevant they will not 'remember' the advertising peer and therefore the peer will be forgotten. Thus, more advertisements are needed to reach the relevant of the related peers, so that they will remember the advertising peers and will send advertisement messages to it too. As we can see, there is no linear decrease in recall and number of advertisement messages. The reason that there are fewer advertisements is that the initial random list of peers is the only set that a peer will know (because it receives no advertisements of peers due to the reason given before) together with the fact that peers are removed from the neighbor list if a messages is sent and

Nr	Simulation setting				$Peer_{prec}$	Doc_{rec}	Q_{msg}	A_{msg}
28	$QF2_{(4,3)}$	$\underline{AI2}_{(0.8)}$	$\underline{AD1}_{(3,5)}$	$\underline{AA1}_{(80)}$	0.0594	0.2181	110	959
29	$QF2_{(4,3)}$	$\underline{AI1}_{(1)}$	$\underline{AD2}_{(3,5,0.2)}$	$\underline{AA1}_{(80)}$	0.1052	0.4036	109	401
30	$QF2_{(4,3)}$	$\underline{AI1}_{(1)}$	$\underline{AD1}_{(3,5)}$	$\underline{AA2}_{(80)}$	0.1792	0.3583	64	1017
31	$QF2_{(4,3)}$	$\underline{AI1}_{(1)}$	$\underline{AD2}_{(3,5,0.2)}$	$\underline{AA2}_{(80)}$	0.1308	0.4383	98	459
32	$QF2_{(4,3)}$	$\underline{AI2}_{(0.8)}$	$\underline{AD1}_{(3,5)}$	$\underline{AA2}_{(80)}$	0.1592	0.4003	78	890
33	$QF2_{(4,3)}$	$\underline{AI2}_{(0.8)}$	$\underline{AD2}_{(3,5,0.2)}$	$\underline{AA2}_{(80)}$	0.1203	0.4215	100	398
34	$QF2_{(4,3)}$	$AI2_{(0.8)}$	$AD2_{(3,5,\underline{0})}$	$AA2_{(80)}$	0.1381	0.4660	95	573
35	$QF2_{(\underline{3,3})}$	$AI2_{(0.8)}$	$AD2_{(3,5,0.2)}$	$AA2_{(80)}$	0.1251	0.2744	40	410
36	$QF2_{(\underline{5,3})}$	$AI2_{(0.8)}$	$AD2_{(3,5,0.2)}$	$AA2_{(80)}$	0.1086	0.4984	198	401
37	$QF2_{(4,3)}$	$AI2_{(0.8)}$	$AD2_{(\underline{2,5,0.2})}$	$AA2_{(80)}$	0.0644	0.2548	107	60
38	$QF2_{(4,3)}$	$AI2_{(0.8)}$	$AD2_{(\underline{4,5,0.2})}$	$AA2_{(80)}$	0.1274	0.4383	95	761

Table 3.5: Semantic query routing with semantic-based advertising policies

Nr	Simulation setting				$Peer_{prec}$	Doc_{rec}	Q_{msg}	A_{msg}
39	$QF2_{(4,3)}$	$AI2_{(0.8)}$	$AD2_{(3,5,0.2)}$	$AA2_{(\underline{10})}$	0.0359	0.1029	72	81
40	$QF2_{(4,3)}$	$AI2_{(0.8)}$	$AD2_{(3,5,0.2)}$	$AA2_{(\underline{20})}$	0.0701	0.2400	90	161
41	$QF2_{(4,3)}$	$AI2_{(0.8)}$	$AD2_{(3,5,0.2)}$	$AA2_{(\underline{40})}$	0.0995	0.3665	99	274
42	$QF2_{(4,3)}$	$AI2_{(0.8)}$	$AD2_{(3,5,0.2)}$	$AA2_{(\underline{60})}$	0.1135	0.4077	99	350
43	$QF2_{(4,3)}$	$AI2_{(0.8)}$	$AD2_{(3,5,0.2)}$	$AA2_{(\underline{160})}$	0.1256	0.4419	100	436
39	$QF2_{(4,3)}$	$AI2_{(0.8)}$	$AD2_{(2,5,0.2)}$	$AA2_{(\underline{10})}$	0.0100	0.0189	53	10
40	$QF2_{(4,3)}$	$AI2_{(0.8)}$	$AD2_{(2,5,0.2)}$	$AA2_{(\underline{20})}$	0.0235	0.0745	78	19
41	$QF2_{(4,3)}$	$AI2_{(0.8)}$	$AD2_{(2,5,0.2)}$	$AA2_{(\underline{40})}$	0.0447	0.1648	97	37
42	$QF2_{(4,3)}$	$AI2_{(0.8)}$	$AD2_{(2,5,0.2)}$	$AA2_{(\underline{60})}$	0.0560	0.2210	103	48
43	$QF2_{(4,3)}$	$AI2_{(0.8)}$	$AD2_{(2,5,0.2)}$	$AA2_{(\underline{160})}$	0.0709	0.2811	111	68

Table 3.6: Influence of number of neighbors

the receiver is off-line, a peer's neighbor list will get smaller and smaller. Experiment 50 shows that one extra advertisement hop solves the problem. We did not perform experiments to find the 'drop-point' for these three hops, but probably it will lie in a network with more than 1M peers. Another solution that partly solves the problem of the enduring reduction of the neighbor list, which we did not test, is to have gateway peers that provide new pointers to random peers when a peer has not enough neighbors anymore.

Influence of availability Table 3.8 shows the results of varying the availability of peers for a pure random query- and advertisement policy and for the semantic-based policies (in the other tables, the availability is size is fixed at 40%, see 4.2). As can be seen, the performance is very dependent on the availability. When the availability is 3%, our semantic-based approach has the same (bad) performance as the random-based approach (compare experiment 51 with 56). However when the availability is 26%, the recall of the semantic-based approach performs already 10 times better than the random approach in terms of recall. It is likely that this difference will even be higher in larger networks because random networks will have a linear decrease in recall when the network grows linear and Table 3.7 shows that the semantic-based approach does not have the linear decrease.

3.7 Conclusions

In this paper we presented a Peer-to-Peer system where the nodes describe their content in terms which occur in a shared similarity matrix. The peers share their content descriptions with other peers where peers remember only those peers that are relevant (i.e. semantically close) to their own content. In this way, peers form a semantic overlay network. We have shown how the model can be applied in a bibliographic scenario based on a realistic data-set. We have shown a method to automatically make a term similarity matrix which can be shared among a group of peers. Simulation experiments that we performed with this bibliographic scenario where all peers share the same distance matrix show that it performs much better in recall and the reduction of the number of messages compared to a random approach. Our results are based on a large data-set and are comparable with the work of [Haase et al., 2004c], except that our shared data-structure is richer and created automatically. We have shown that our system is scalable and robust to peer drops, although the availability of peers should not be too low. Future work has to solve the problem of dealing with very low availabilities, most likely with the introduction of a complementary protocol to renew neighbor caches.

Nr	Nr_p	Simulation setting	$Peer_{prec}$	Doc_{rec}	Q_{msg}	A_{msg}
44	5K	$QF2_{(4,3)}AI2_{(0.8)}AD2_{(2,5,0.2)}AA2_{(80)}$	0.0601	0.3895	99	36
45	10K	$QF2_{(4,3)}AI2_{(0.8)}AD2_{(2,5,0.2)}AA2_{(80)}$	0.0576	0.3157	106	44
46	20K	$QF2_{(4,3)}AI2_{(0.8)}AD2_{(2,5,0.2)}AA2_{(80)}$	0.0609	0.2433	105	55
47	27K	$QF2_{(4,3)}AI2_{(0.8)}AD2_{(2,5,0.2)}AA2_{(80)}$	0.0586	0.2125	102	58
48	40K	$QF2_{(4,3)}AI2_{(0.8)}AD2_{(2,5,0.2)}AA2_{(80)}$	0.0584	0.1682	96	61
49	80K	$QF2_{(4,3)}AI2_{(0.8)}AD2_{(2,5,0.2)}AA2_{(80)}$	0.0265	0.0374	60	24
50	80K	$QF2_{(4,3)}AI2_{(0.8)}AD2_{(3,5,0.2)}AA2_{(80)}$	0.1092	0.3497	107	538

Table 3.7: Influence of the maximum number of peers in the network for our best performing settings

Nr	α	Simulation setting				$Peer_{prec}$	Doc_{rec}	Q_{msg}	A_{msg}
51	3%	$QF1_{(4,3)}$	$AI1_{(1)}$	$AD1_{(3,5)}$	$AA1_{(80)}$	0.0105	0.0083	12	32
52	13%	$QF1_{(4,3)}$	$AI1_{(1)}$	$AD1_{(3,5)}$	$AA1_{(80)}$	0.0080	0.0313	89	392
53	26%	$QF1_{(4,3)}$	$AI1_{(1)}$	$AD1_{(3,5)}$	$AA1_{(80)}$	0.0080	0.0331	116	757
54	40%	$QF1_{(4,3)}$	$AI1_{(1)}$	$AD1_{(3,5)}$	$AA1_{(80)}$	0.0072	0.0281	124	1099
55	60%	$QF1_{(4,3)}$	$AI1_{(1)}$	$AD1_{(3,5)}$	$AA1_{(80)}$	0.0072	0.0242	123	1442
56	3%	$QF2_{(4,3)}$	$AI2_{(0.8)}$	$AD2_{(3,5,0.2)}$	$AA2_{(80)}$	0.0169	0.0084	11	14
57	13%	$QF2_{(4,3)}$	$AI2_{(0.8)}$	$AD2_{(3,5,0.2)}$	$AA2_{(80)}$	0.0563	0.1705	72	105
58	26%	$QF2_{(4,3)}$	$AI2_{(0.8)}$	$AD2_{(3,5,0.2)}$	$AA2_{(80)}$	0.0933	0.3803	100	240
59	40%	$QF2_{(4,3)}$	$AI2_{(0.8)}$	$AD2_{(3,5,0.2)}$	$AA2_{(80)}$	0.1203	0.4215	100	398
60	60%	$QF2_{(4,3)}$	$AI2_{(0.8)}$	$AD2_{(3,5,0.2)}$	$AA2_{(80)}$	0.1483	0.4422	96	521

Table 3.8: Influence of peer availability

Chapter 4

pNear

This chapter is based on the following papers: [Siebes, 2005], [Siebes,]

Discovering relevant nodes according to a query is a challenging problem in Peer-to-Peer (P2P) systems. Currently two promising directions to solve this problem are (1) distributed hash-tables (DHTs) and (2) semantic overlay networks (SONs) which again can be divided into systems that cluster peers with similar content based on term overlap and systems that map both the content and queries on a shared semantic data structure. In this paper we present the pNear system that combines DHTs with clustering via term overlap and show that we are able to tackle some important disadvantages that hold for the individual approaches. We evaluate our approach via simulations based on a large and realistic data-set that we have constructed for this purpose, and which may be useful for similar experiments by others. Furthermore, pNear has some potentially beneficial semantic functional side-effects, which we call term semantics. Namely, it could be able to efficiently (1) give real-time feedback on determining if a term is very generic (e.g. 'computer_science') or more specific (e.g. 'description_logics') and (2) give insight if a term is popular (e.g. 'datamining') or infrequently used (e.g. 'prime_computing').

4.1 Introduction

Undisclosed content, lack of privacy and the possibility to censor data are seen as important disadvantages of the centralized approach of today's popular search engines. Firstly, in such a centralized approach, the owner of

the server has complete control over which content and in which order the content is presented to the user. The drawback of this approach is that authorities could force the search engine not to show some content that they do not like. Secondly, much data on the web is dynamically generated via databases and therefore are very difficult to crawl by search engines. Often this is by intention of the provider because it wants a unique access point for users to find its data, guaranteeing user traffic to the web-site and/or keeping full control over the data. Thirdly, also privacy is an important issue that is in principle not guaranteed by centralized search engines. Namely, search engines easily can connect IP-addresses with queries and can make a profile of the users behind it. Peer-to-Peer systems, where nobody is in control, are in principal much more difficult to be used for tracing the behavior of users. These three issues are important reasons for doing research on P2P-based search engines.

A big advantage of centralized search engines is that the number of messages needed in the query process often is only 2 and the number of hops is only 1, guaranteeing efficient bandwidth usage and quick response times. In Peer-to-Peer systems, the number of messages and hops mainly depends on how quickly the relevant peers are found. Needless to say that much of the current research on P2P-based document searching is focussed on reducing the number of messages and hops to generate an attractive alternative to the centralized approach especially when privacy, undisclosed content and censorship play a role.

In this paper we propose a Peer-to-Peer system named pNear, where peers describe their content by a set of terms. pNear combines distributed indexing based on Distributed Hash Tables (DHT) together with clustering peers with similar content descriptions without depending on a shared data-structure like in pSearch [Tang et al., 2002] and Bibster (described in Chapter 2) or a complete distributed index like in GridVine [Aberer et al., 2004]. We show that only a small part of the terms in the content descriptions need to be indexed via DHT to still guarantee that most peers that contain the asked terms ar found, even when the querying peer is not clustered due to an empty expertise description or posts a query that is not related to its expertise. Once a set of matching peers is found for a query by the pNear system, algorithms provided by research in Information Retrieval allow to order this set of peers according to their relevance to the query. Note that these algorithms are applied *after* the set is found and therefore not interfere with the pNear algorithm. This means that ranking is a different independent topic which is not covered in this paper. The next section provides a brief overview of existing work on DHTs and semantic overlays. Section 3 shows our model that combines methods that are described in section 2. Section 4 presents an empirical validation via simulation experiments. Section 5 concludes the work.

4.2 DHT's and Semantic Overlays

4.2.1 DHT

Distributed hash tables (DHTs) are currently seen as an important building block for peer-to-peer systems for storing and allocating content in a completely decentralized way [Aberer et al., 2003, Ratnasamy et al., 2001, Stoica et al., 2001, Rowstron and Druschel, 2001]. This allows each node to function independently and collectively form the complete efficient search system without any central coordination. The general idea of DHTs is that each item shared on the network is hashed to a unique key, and that this key together with the content (or a pointer to it) are efficiently routed to the unique peer responsible for that key. In this way, each peer is responsible for storing the content (or a pointer to it) that is associated with the key. In principle, all DHT based systems provide two functionalities: $store(key, object)$ storing an $object$ identified by its key, and $search(key)$ which returns the $object$ (when it exists) from the peer whose network identifier is numerically closest to the key. The current systems based on DHTs provide these efficient key lookup and storage algorithms needing only $O(log(N))$ messages per search and storage, where N is the number of peers in the network.
There are also some disadvantages with standard DHT approaches:

- Firstly, there are the administration costs needed to maintain the network overlay during content updates and at peer joins, leaves and failures. This results in much maintenance traffic over the network when there are many updates or when peers frequently join and leave the network. Aberer et al. [Aberer et al., 2003] confirms this problem and provide a more scalable DHT update mechanism.
- Secondly, when the data distribution is extremely skewed, for example in document distributions on the web that follow a Zipf-distribution, additional load-balancing algorithms on top of DHT are needed to equally distribute data over the network to prevent node bottlenecks. The work of Byers et al. [Byers et al., 2002] confirms the load-balancing problem and describes an alternative DHT approach to solve it, by paying the price of significant additional maintenance costs.
- Thirdly, in standard DHT approaches each key is mapped to one peer, which means that when this peer for whatever reason does not respond, the content cannot be found on the network.

4.2.2 Semantic Overlays

Peers that know about the content of other peers form a Semantic Overlay Network (SON). Edutella [Nejdl et al., 2002] is a schema based network

where peers describe their functionality (i.e. services) and share these descriptions with other peers. In this way, peers know about the capabilities of other peers and only route a query to those peers that are probably able to handle it. Although, Edutella provides complex query facilities, it has still no sophisticated means for semantic clustering of peers, and their broadcasting does not scale well. Gridvine [Aberer et al., 2004] uses the semantic overlay for managing and mapping data and metadata schemas, on top of a physical layer consisting of a structured peer-to-peer overlay network, namely P-Grid, for efficient routing of messages. In essence, the good efficiency of the search algorithm is caused not by clustering of semantically related peers based on the semantic overlay, but by efficient term storage and retrieval characteristics of the underlying DHT approach for mapping terms to peers.

Another SON approach is to classify the content of a peer into a shared topic vector where each element in the vector contains the relevance for that given peer for the respective topic. pSearch [Tang et al., 2002] is such an example where documents in the network are organized around their vector representations (based on modern document ranking algorithms) such that the search space for a given query is organized around related documents, achieving both efficiency and accuracy. In pSearch each peer has the responsibility for a range expressed by each element in the topic vector, e.g. $([0.2-0.4],[0.1-0.3])$. Now all expertise vectors that fall in that range are routed to that peer, meaning that, following the example vector, the expertise vector $[0.23, 0.19]$ would be routed to this peer and $[0.13, 0.19]$ not because 0.13 does not fall in between 0.2 and 0.4. Besides the responsibility for a vector range, a peer also knows the list of neighbors which are responsible to vector ranges close to itself. The characteristic of pSearch is that the way that peers know about close neighbors is very efficient. A disadvantage of pSearch is that all documents have to be mapped into the same (low dimensional) semantic search space and that the dimensionality on the overlay is strongly dependent of the dimensionality of the vector, with the result that each peer has to know many neighbors when the vectors have high a dimension.

Another approach is based on random walk clustering, where peers with similar content are going to know each-other [Voulgaris et al., 2004]. The assumption is that queries posted by (the users of) peers are semantically closely related to the content of the peer itself. This results in a high probability that the neighbors of the peer (the peers in the cluster of that peer) have answers to the query. The problem of this approach in the domain of full-text searches, is what information a peer has to tell to another peer so that they are able to determine if they are related or not. When there is no shared data-structure (like a fixed set of terms) in which they can describe their content, the whole content has to be shared. This results in the fact that much data has to be shared between peers for determining closeness. In

[Sripanidkulchai et al., 2003], peers cluster passively by remembering peers that previously provided them with satisfying answers using a Gnutella overlay. This means that for each peer, clustering is achieved after an initial 'warm-up' period, which could take a while. In terms of performance, they achieve a workload reduction by a factor of 3 to 7 in comparison to Gnutella. The system described in [Cohen et al., 2003] uses a similarity measure based on keyword overlap between the set of terms in a query and the set of terms that describe the content of a peer. Unfortunately, in the paper there is no information about the clustering overhead in the number of additional messages to form the overlay, which makes it difficult to compare with other approaches.

In contrast to the previous approach, the last SON approach that we discuss here lets peers describe their content in a shared set of terms. Mostly these terms are organized in a topic network or hierarchy making it able to determine the semantic similarity between terms. Each peer is characterized by a set of topics that describe its expertise. A peer knows about the expertise topics from other peers by analyzing advertisement messages [Haase et al., 2004c] or answers [Tempich et al., 2004]. In this way peers form clusters of semantically related expertise descriptions. Given a query, a shared distance metric allows to forward queries (described by a shared set of terms) to neighbors of which their expertise description is semantically closely related to the query. The advantages of the SON approaches are threefold:

- *Peer autonomy* Each peer can, in principle, have its own distance measure, peer selection mechanism and clustering strategy. For example, this allows peers to keep their neighbor list or similarity metric secret. Also peers can decide at any time to change their visibility on the network by sending advertisement messages.
- *Automatic load balancing* When some content is provided by many peers also the semantic cluster on that content will contain many peers. In this way, load balancing is an emergent property of this approach.
- *Robustness/fault tolerance* When peers leave the network or do not respond to a query, the only consequence is that they probably will not be asked a next time until they send new advertisement messages or are recommended by other peers. In contrast, most DHT approaches have to move routing tables to other peers in order to restore the overlay.

Most SON approaches have also one or both following disadvantages:

- Firstly, some SON approaches rely on a shared data-structure (distance matrix, topic-hierarchy or term-vector) in which the content has to be described. When a peer has content which cannot be classified into this data-structure, it has to be extended. This change than has to be sent to all peers in the network which generated much network traffic.

- Secondly, many SONs rely on the assumption that the queries of a peer are related to its own content and that a peer also has content that allows it to cluster itself into the overlay. These assumptions are not always realistic.

In the next section we propose an approach that combines DHT with SON which benefits from the advantages of both approaches but where the combination does not suffer from the individual disadvantages.

4.3 Combining Expertise Clustering and Distributed Indexes

In this section we describe our approach by first giving an informal description of the registering process and query process and later on more precisely by providing a formal description of the elements of the system and the algorithms.

4.3.1 Informal description of pNear

Registering In pNear, besides sharing content, peers play two additional roles, namely that of *expertise cache* and of *expertise register*. The word 'expertise' should be read as a set of terms that describe the content from a peer together with its network identifier. Expertise registers are distributed indexes that map terms to peers that registered themselves as experts on these terms. The registering process is as follows: when a peer joins the network and/or has new content, it summarizes the content that it shares by a set of terms that we name *expertise descriptions*. After that, a small random set of terms from the expertise descriptions are selected for registering where each term from this set is hashed to a unique key that serves as the identifier of the *register message* that has to be routed, via DHT algorithms, to the peer (register) that is responsible for the key. These register messages each contain the term that was hashed, the sending peer and its expertise description. In this way a register which is responsible for a given term t contains a set of peers with their expertise descriptions that registered themselves on the term t. The process of storing and retrieving content mapped to keys is very efficient, because it is based on the DHT algorithms [pastry, chord, can] that only need $O(\log(N))$ messages (where N is the number of peers in the network) to retrieve or store a key and the mapped content. The registering process is not only meant for distributing expertise descriptions to registers so that they can be used in the query process, but also to allow the registering peers to know other peers that registered themselves at the register on the same terms. Namely, when a register message is processed by a register,

it returns a *register result message* containing pointers to peers that are similar together with their rankings based on a shared similarity measure on the expertise descriptions. Now that the registering peer has some pointers to related peers, the clustering process starts. Namely, the peer sends an *advertisement message* to some of the peers (which were returned by the register) containing the senders expertise description. In this way the receiver gets to know the sender and when its expertise is semantically similar, it stores the peer in its *expertise cache*. Expertise caches therefore will contain pointers to peers that have similar content to the peer that maintains the cache. Note that expertise caches are different from registers because caches store descriptions from peers that are relevant for the host of the cache and registers contain descriptions of peers that fall into the responsibility of the DHT key range of the host. It also looks in its own expertise cache for peers that are semantically close to the sender of the advertisement message. If some peers are found, they are put together with their rankings in an *advertisement result message* and sent back to the initiator of the advertisement message. In this way also the sender gets to know even more peers that have similar content to which it can repeat the advertising process. This process stops till a given maximum number of *advertisement rounds*.

Querying The query process is started when a user initiates a query on the peer that represents that user on the network. First, the peer abstracts the query, e.g. via stemming and removing stop-words, into a set of terms that is used to compare the similarity between the query and expertise descriptions that it and others will encounter in the query distribution process. Next, the peer looks in its local expertise cache for peers whose expertise descriptions are semantically close to the terms in the query abstraction. When there are enough peers these are selected to send the *query messages* to, containing the query itself and the query abstraction. The receiving peer tries to answer the query and looks in its own expertise cache if it knows some peers that are semantically close to the query abstraction and sends both the eventual answer and the eventual set of pointers to related peers back via a *query result message*. When the initiator of the query receives the messages it (or the user) decides whether to continue the query process or whether it is satisfied with the number of answers. It could be that the peer that initiated the query has no, or not enough, semantically related peers in its expertise cache to send the query to. This could for example happen when the peer does not share any content on the network, resulting in no expertise description, or when the user posts a query that is completely unrelated to the content (and therefore the peers in the cache) that a peer shares. When this happens, the expertise registers come into place. Namely, first the terms in the query abstractions are hashed to unique keys that will serve as identifiers of the *register consult messages* that the peer will send to the registers. A register consult message contains the term that was hashed, the senders identifier

and the complete query abstraction. The routing process of these messages is identical to the expertise registering process.

4.3.2 Elements of the system

Expertise description Each peer has an expertise extraction function ϵ : $docs_p \mapsto \mathcal{T}_p$ which extracts a set of terms $\mathcal{T}_p \subset 2^{\texttt{STRING}}$ from the peer's documents $docs_p$ it wants to share in the network. These terms describe what the content of the peer's documents is about. This function could be based on an automatic NLP algorithm like GATE [Cunningham et al., 1997] or be performed by a user that selects a set of terms which classifies the documents.

Query abstraction For simplicity reasons we assume that a query posted by a user is just a set of words, thus without Boolean operators or other constructs except an implicit AND between the words. An answer on the query is therefore correct when all the words are in each of the result documents. The words in the query are mapped by a mapping function π : *query* $\mapsto \mathcal{T}_q$ into a set of terms $\mathcal{T}_q \subset 2^{\texttt{STRING}}$. These terms could exactly match the terms in the users query, but also be stemmed terms and where stop-words are removed.

Similarity measure The registers and caches in our P2P network each use the same similarity measure to calculate the similarity between two expertise descriptions or between an expertise description and a query:

$$\sigma : (\mathcal{T}_p, \mathcal{T}_p \bigcup \mathcal{T}_q) \mapsto \mathbb{R}^+$$

where $\mathcal{T}_p$ consists of sets of terms from expertise descriptions and $\mathcal{T}_p \bigcup \mathcal{T}_q$ sets of expertise terms (i.e. expertise descriptions of peers) or abstracted query terms. In our simulations we instantiate σ in the following way:

$$\sigma(\bar{t}_p, \bar{t}_{p,q}) = \sum_{t_i \in (\bar{t}_p \bigcap \bar{t}_{p,q})} generality(t_i, R)^\alpha \tag{4.1}$$

where $generality(t_i, R) \in [0, 1]$ gives the number of expertise descriptions in the register R that contains the term t_i divided by the total number of expertise descriptions in R. This number is a measure of the discriminating power of the term compared to the other terms. If the term occurs in all expertise descriptions, it is less discriminating. The α parameter determines the influence of this generality in the similarity measure. In our simulations we use for the registers and caches two different values for α which we discuss later in the section on the experimental setup.

Expertise register Each peer in the network can be responsible for one or more expertise registers. A peer has the responsibility for those registers of which the hash key of the term (the term for which the register keeps the registered peers) falls into the responsibility of that peer. Each expertise register contains a set of $ID \times \mathcal{T}$ tuples, containing peer identifiers and their expertise descriptions. These tuples all were routed via a DHT algorithm to this register based on a unique key that was hashed from the term that the sender selected for registering. This means that each register is only responsible for one single term. However, a peer can maintain more than one register. To guarantee an equal distribution of registers over the peers in the network, one should use a statistically random key generator like the MD5 algorithm [Rivest, 1992]. When a key is routed to the peer whose network identifier is closest to the key, it could be that there does not exist a register on that peer for the term. In that case the register is created on the fly. Each register has a maximum number of expertise descriptions that it can store. When this maximum is reached a new description replaces an existing one. For now we leave it up to the DHT algorithms to deal with moving and duplicating the register information to deal with node drops. In our simulation experiments we assume a stable network without node drops. Although it is future work to show how our system responds on an instable network, we can already know that it will be much more stable than a pure DHT approach. This is because when a peer in a pure DHT network responsible for a key does not respond, all the queries on that key cannot be answered. In contrast, a SON is more robust because the peer that queries has a high probability that its neighbors are able to answer the query.

Expertise cache Like an expertise register, an expertise cache also contains a set of $ID \times \mathcal{T}$ tuples, containing peer identifiers and their expertise descriptions. However in this case the expertise descriptions are semantically close to the expertise description of the peer itself. This set is the result of a clustering process that we describe later. When the maximum number of expertise descriptions that an expertise cache can store is reached, a new description is compared with the least similar description, and if the new one is closer to the peers own content, it replaces the least similar one.

Register algorithm The registering process which allows a peer to advertise its expertise and to discover similar peers is described in algorithm 2. First, a peer selects a set of terms from its expertise description (step 6) and for all these terms it calculates the hash code (step 9). The DHT protocol takes care that for each register message for each selected term (with the hash-key as identifier) the right register is found. In step 10, each found register stores the sender's expertise description and returns a set of ranked peers. From step 17-23, an iterative process of selecting the best ranked

peers to advertise expertise is performed, where each receiving peer possibly returns an extra set of ranked peers which is included in the 'ranked-peer' set. The maximum number of messages used for the registering process (i.e. building the semantic overlay) is by taking the sum of two types of messages: first the messages needed to route the expertise descriptions to the right registers via the DHT overlay for a subset of terms from this expertise description and secondly the advertisement messages that a peer sends to related peers that are returned by the registers and by each advertisement round.
The number of messages needed for routing can be calculated by multiplying the number of terms that are selected for registering times the number of messages needed to store each term (a key) in a DHT overlay. Although this differs per DHT implementation, taking the $\log$ of the number of peers in the network is a good and bit pessimistic estimate: $O(\#Terms \times^2 log(\#Peers))$. The number of messages needed for advertising can be calculated by simply multiplying the number of advertisement rounds with the number of neighbors selected per round: $O(\#Rounds \times \#Neighbors)$. To make this more clear, we give the following example: when a peer selects 3 terms (out of the on average 127 terms from an expertise description), this results in $3 \times^2 log(100,000) \approx 50$ direct messages needed to store the object in the DHT overlay. When a peer for the advertisement process uses 5 rounds and selects 4 neighbors per round, 20 (5×4) messages are needed for the advertisement process. Therefore, in this case a peer maximally around $50 + 20$ messages are used. In practice this number will be a bit lower because a peer has not always enough neighbors to select.

Query algorithm The query process is described in algorithm 3. As already described in the informal description, a peer that is already clustered in the network perhaps does not need to consult the registers to find pointers to the cluster. However, in our experiments we make it purposely harder by only letting unclustered to do queries which means that always the registers need to be consulted. Therefor, in step 6 of the algorithm we iterate over all the query terms, which means that every register responsible for a term in S_q is queried. For each step 8, a reached register returns a ranked set of peers for the given query. From step 15 to step 21, an iterative process selects the best set of peers from the $rankedPeers$ In each step 18 the best peer from the $rankset$ is queried and possibly returns (if it is online and has answers) a set of documents ($docs$) that match the query q and a ranked set of peers ($returnedRankedPeers$) which is joined with the $rankedPeers$ set (step 19). This process ends when the user terminates the process (when (s)he is satisfied) or when the maximum number of query rounds (step 10) is reached.

The maximum number of messages used per query posted by a peer depends on how well a peer is clustered in the network and how related the query is

to the peer's expertise description. When the peer is well clustered and posts related queries, the chance that its neighbors can answer the query is higher than when a peer has no expertise description and therefore is not clustered or posts a query which is unrelated to its expertise. In the first case a peer can decide not to query the registers but immediately query its neighbors. In our experiments we simulate the worst case scenario (in terms of search complexity) where the peers that query are not clustered in the network, which means that in our experiments each peer always needs to contact the registers to find pointers to the right clusters. To find these registers, the number of messages needed is the number of terms per query multiplied by the number of messages needed to find the corresponding register in the DHT overlay, which is dependent on the DHT protocol that is used, but again about taking the $\log$ of the number of peers in the network. When these registers are found and have returned a set of recommendations, the remaining number of messages that are used in the query process can be calculated by multiplying the number of query rounds by the peers selected per round.

4.3.3 Expected behavior

Now that we have described our method we now describe how it circumvents the drawbacks of the individual methods where it is based on. We do this by showing results of simulation experiments in the following section combined with the following argumentation:

The drawbacks of pure DHT again are first its expensive maintenance, where there is a linear increase of costs with the number of terms that are stored in the network. The results of the next section have to show that we can reduce the number of terms to only a fraction of the original set. The experiments also need to show that the maintenance costs of the semantic overlay are much less than the maintenance of the pure DHT overlay. Secondly in a pure DHT approach only one peer is responsible for a key space. When this peer contains a very popular key, the peer may become a bottleneck if it has not enough bandwidth or processing power to deal with the huge number of requests. Although in our system there is still one register for each hashed term, the clustering of related peers allows that, given the assumption that interest and expertise are related, even when the register does not respond, the neighbors of the querying peer are able to answer it and/or know some good candidates. Note that the assumption only is needed in this case where a register does not exist or respond. Thirdly the skewed distribution of content and queries result that in pure DHT some peers have to store much more keys and deal with much more queries than other peers. Results need to show that our method reduces this problem because the method allows peers to have a small fixed register and cache size and still give good results. The main drawbacks of semantic overlays are that either they need a shared se-

Algorithm 2 $register(p)$: Registering process of a peer p

1: Let $E_p \in E$ be the expertise description of p
2: Let id_p be the identifier of p in the network
3: Let $nrOfSelections$ be the system parameter dictating the number of terms that should be randomly selected for the registering process
4: Let $rankedPeers := \emptyset$ and $returnedRankedPeers := \emptyset$ both be initially empty sets of tuples $2^{[ID,Rank]}$ containing peer id's and their similarity rank ($Rank \in \mathbb{R}^+$) to E_p
5: Let h be the hashed representation of a term
6: Let $S_e := getRandomSubset(E_p, nrOfSelections)$ be a random subset of n terms ($S_e \subseteq E_p$) that will be used for registering by p
7: $rankedPeers := sortCacheOnSimilarity(C_p, E_p)$ `// place the peer identifiers from the local cache Cp in rankedPeers together with their ranks according to the similarity on the expertise description Ep`
8: **for all** $s \in S_e$ **do**
9: $h := hash(s)$ `//generate unique hash key as message identifier`
10: $returnedRankedPeers := routeRegMsg(h, id_p, E_p)$ `//route the message via DHT on the key and assign the answer of the register to returnedRankedPeers`
11: $rankedPeers := rankedPeers \bigcup returnedRankedPeers$
12: Let $maxRegRounds$ be a system parameter indicating the maximum number of clustering rounds
13: Let $maxRegVisitsPerRound$ be a system parameter indicating the maximum number of visits per round
14: Let $visitedPeers := \emptyset$ be the initially empty set of peers that are visited during the current registering process
15: Let $cacheSize$ be a system parameter dictating the maximum number of items in the peer's expertise cache
16: Let $C_p \subset 2^{[ID,E]}$ be the expertise cache of p containing a set of tuples with peer identifiers and their expertise descriptions
17: **for** $1 : maxRegRounds$ **do**
18: **for** $1 : maxRegVisitsPerRound$ **do**
19: $id_c := takeBestPeer(rankedPeers, visitedPeers)$ `//select the peer with the highest rank which is not in visitedPeers`
20: $[E_c, returnedRankedPeers] := routeAdvMsg(id_c, id_p, E_p)$ `//send the advertisement message directly to c where it stores p's expertise description and returns c's expertise description and a ranked list of semantically related peers.`
21: $rankedPeers := rankedPeers \bigcup returnedRankedPeers$
22: $C_p := includeExpertiseDescription(C_p, id_c, E_c, cacheSize)$
23: $visitedPeers := visitedPeers \bigcup \{id_c\}$

Algorithm 3 $query(q, p)$: query q is initiated by the user of peer p

1: let S_q be the set of terms that abstract the query q as described in the previous paragraph
2: Let $visitedPeers := \emptyset$ be the initially empty set of peers that are visited during the current querying process
3: Let $rankedPeers := \emptyset$ and $returnedRankedPeers := \emptyset$ both be initially empty sets of tuples $2^{[ID,Rank]}$ containing peer id's and their similarity rank ($Rank \in \mathbb{R}^+$) to S_q
4: Let $C_p \subset 2^{[ID,E]}$ be the expertise cache of p containing a set of tuples with peer identifiers and their expertise descriptions
5: $rankedPeers := sortCacheOnSimilarity(C_p, S_q)$ `// place the peer identifiers from the local cache Cp in rankedPeers together with their ranks according to the similarity on the query terms Sq`
6: **for all** $s \in S_q$ **do**
7: $h := hash(s)$ `//generate unique hash key as message identifier`
8: $returnedRankedPeers := routeRegConsultMsg(h, S_q)$ `//route the message via DHT on the key and assign the answer of the register to returnedRankedPeers`
9: $rankedPeers := rankedPeers \bigcup returnedRankedPeers$
10: Let $maxQueryRounds$ be a system parameter indicating the maximum number of query rounds
11: Let $maxQueryVisitsPerRound$ be a system parameter indicating the maximum number of visits per round
12: Let $visitedPeers := \emptyset$ be the initially empty set of peers that are visited during the current registering process
13: Let $cacheSize$ be a system parameter dictating the maximum number of items in the peer's expertise cache
14: Let $C_p \subset 2^{[ID,E]}$ be the expertise cache of p containing a set of tuples with peer identifiers and their expertise descriptions
15: **for** $1 : maxQueryRounds$ **do**
16: **for** $1 : maxVisitsPerQueryRound$ **do**
17: $id_c := takeBestPeer(rankedPeers, visitedPeers)$ `//select the peer with the highest rank which is not in visitedPeers`
18: $[docs, returnedRankedPeers] := routeQueryMsg(id_c, id_p, S_q, q)$ `//send the query message directly to c where it locally searches for answers (`$docs$`) on` q `and tries to find a ranked list of peers that are semantically related to` S_q`.`
19: $rankedPeers := rankedPeers \bigcup returnedRankedPeers$
20: $showResultsToUser(docs)$ `//present the documents that were returned to the user`
21: $visitedPeers := visitedPeers \bigcup \{id_c\}$

mantic data structure which is expensive to update and can be very large or it completely depends on the assumption that peers have an expertise description and that the expertise of a peer and its interests (the queries) are closely related. Our method does not depend on a shared semantic data-structure and in our experiments we let queries originate from peers that are not part of the semantic overlay and which also have no expertise description from themselves.

4.4 Experimental setup

In this section we show how we evaluate our approach and the results by describing the data set, determining the evaluation criteria and showing the results of our simulations.

4.4.1 Data set

In this subsection, we describe our data-set which allows us to simulate queries on real web-pages as they appear on a real (centralized) search engine, and measure how these realistic queries on realistic data-sets perform in a distributed P2P setting instead. We believe that the creation of this data-set in itself is a contribution of this paper. The data-set is available from the author on request. We built our data set in the following way:

- **Query set** Q We used SearchSpy[1] to crawl a set of real user queries. In this way we get a realistic cross-section of popular queries. SearchSpy offers the possibility to filter out queries that are 'family unfriendly', which more or less means that the queries on porn are removed from the set. We chose to use this filter. The crawling process resulted in a set of 28.606 unique user queries Q with an average length of 2.88, a minimum of 1 and a maximum of 15 terms per query. The distribution of the number of terms per query is shown in Figure 4.1, where, for readability reasons queries that contain more than 9 terms are not shown. Future work is to play with other distributions.
- **Pointers to web-pages** G For each query that we crawled we used Google[2] to find at least 1 and at most 100 web-pages in the English language that match q and put the URL's in a set G. In this way, we get via Google a set of popular web-sites. When a query has not a matching web-page, the query is ignored which happened in 0.06% of the queries. This resulted in a set of 28.589 queries that have on average 84 URL pointers per query.

[1] http://www.infospace.com/info.xcite/searchspy
[2] http://www.google.com

- **Terms extractions from web-pages** T For each $g \in G$ we crawled the textual content of the web-page that belongs to the corresponding URL. For each crawled web page we used a natural language processing tool, called TextToOnto [Maedche and Staab, 2001], which extracted the terms that occur at least three times (indication that term is important) in the documents, removed the stop words and stemmed them. Documents of which no terms could be extracted are removed from the data set which happened in 0.4% of the cases. The crawling process resulted in a set of average number of terms per document is 127, the minimum is 1 and the maximum is 2184. For readability reasons we only partly show the distribution of terms per document in Figure 4.2.
- **Simulation Queries** Q_s Instead of using Q, we choose to select random subsets of minimally i and maximally j (both parameters in our simulation) terms from T that will function as queries Q_s that peers send during the simulation experiments. The reason for creating this artificial set is twofold.

  Firstly, it gives us more flexibility to indirectly play with the number of matching peers that are in the network, namely queries that contain many terms normally have less answers than short queries. In other words, we can change distribution of matching peers. In our experiments we will choose a distribution with on average only a few matching peers per query, which makes it 'harder' for our system to find the answers. The distribution of the number of matching peers per query is shown in Figure 4.3, which is the same for all simulation experiments. As can be seen, most of the queries have only a few matching peers, which makes the chance that a peer by coincidence finds the matching peers very small. The average number of peers that match a query is 34, but due to the exponential curve most queries have a smaller number of matches and only some queries have many matching peers.

  Secondly, we are now able to create many queries for each data-set. For example, for an experiment with 10K peers, we generate 3 times 20K random queries (three simulations per setting). In other words, our system is now able to test different data-sets with any number of queries without needing this query-set in advance.

4.4.2 Evaluation criteria

We explore the characteristics of our system by using the peer recall as an evaluation criterium. We think that recall is an accurate measure of user satisfaction, because it gives an indication of how many peers are found

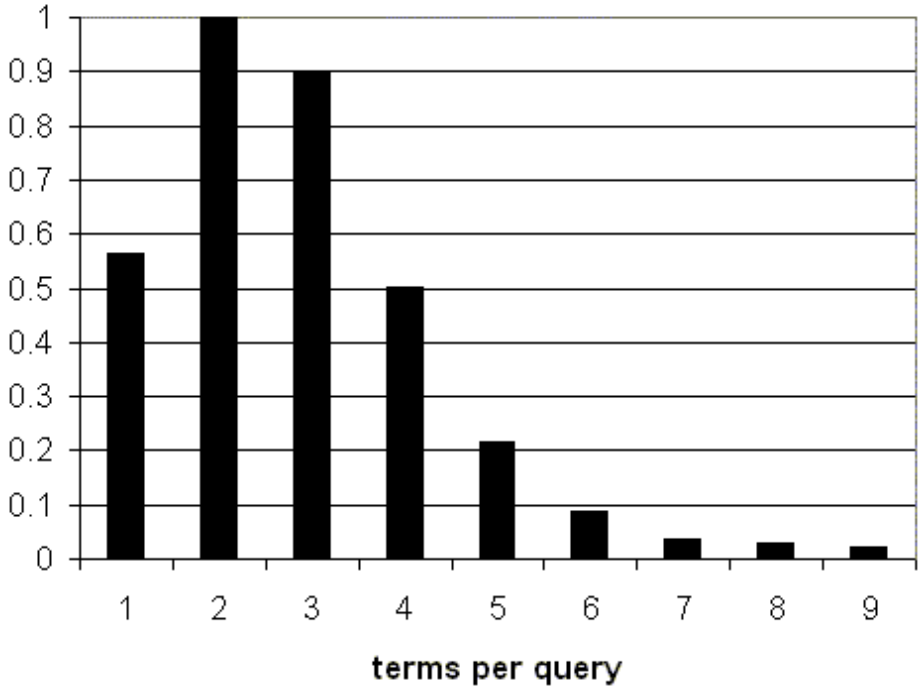

Figure 4.1: The distribution of the average number of terms per crawled query.

during a query process whose expertise descriptions matched all the terms in each user query. The peer precision (fraction of matching peers and visited peers) can be derived by looking at the steepness of the curves that show the recall per query round (shown later in this document). Given that on average there are 34 peers that match for each query and the number of query messages per round is limited to a fixed maximum, it can easily be calculated what the precision is. For example, when in the first round the recall was 20%, then on average 7 peers are found in the first round. Given that the maximum number of query messages in that round is for example 20, then the precision is 7/20 $\times$ 100%. This also means that when the curve is almost horizontal, there is no increase in recall which indicates that almost no new matching peers are found resulting in a low precision. The reason for that could be that most peers are already found.

- **Peer recall per round** Rec_{round}

$$\frac{|P_{total\_relevant} \bigcap P_{queried\_relevant}|}{|P_{total\_relevant}|} = \frac{|P_{queried\_relevant}|}{|P_{total\_relevant}|}$$

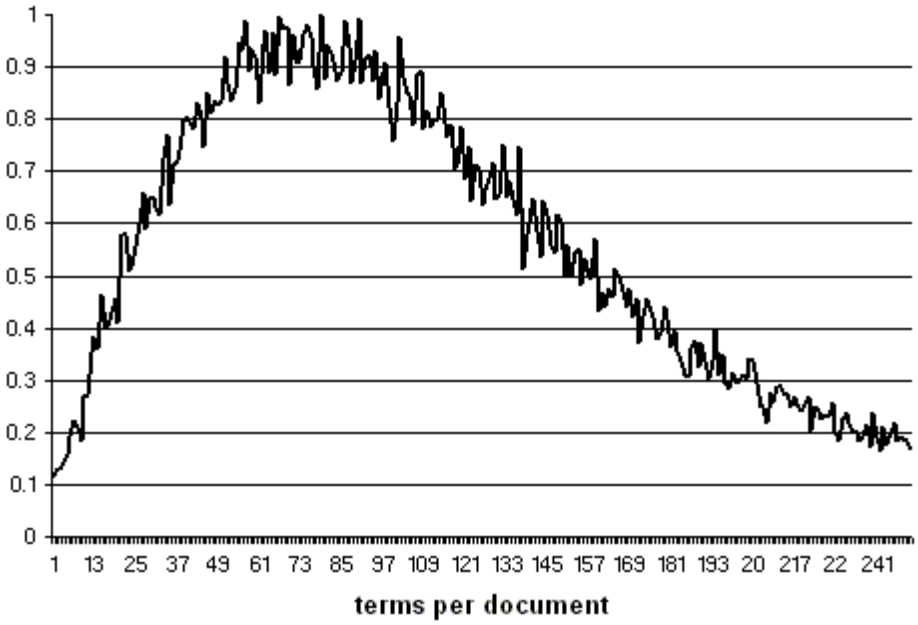

Figure 4.2: The distribution of the average number of terms per expertise description.

The peer recall per round states the cumulative fraction of the relevant peers in the network $P_{queried\_relevant}$ that are queried $P_{queried\_relevant}$ until the current query round $round$. This measure can be seen as an indication for user satisfaction. As we will see, the optimal policies differ for distinct recall levels which means that the optimal settings of the network depends on the recall requirements from the users.

4.4.3 Simulation platform

We wrote an efficient simulation tool in JAVA that allows to simulate a network of more than half a million peers based on the default settings given in Table 4.2. An experiment with 400K peers needs 2.8GB of RAM and 11 hours of CPU time on a Sun UltraSPARC-IIIi 1281 MHz CPU with 16.0GB of RAM. We used the Primitives Collections for Java (PCJ) library[3] which allowed very efficient memory and fast storage/lookup mechanisms needed to search and store the expertise descriptions in the caches and registers, and to match the terms in the queries with the expertise descriptions.

[3]http://pcj.sourceforge.net

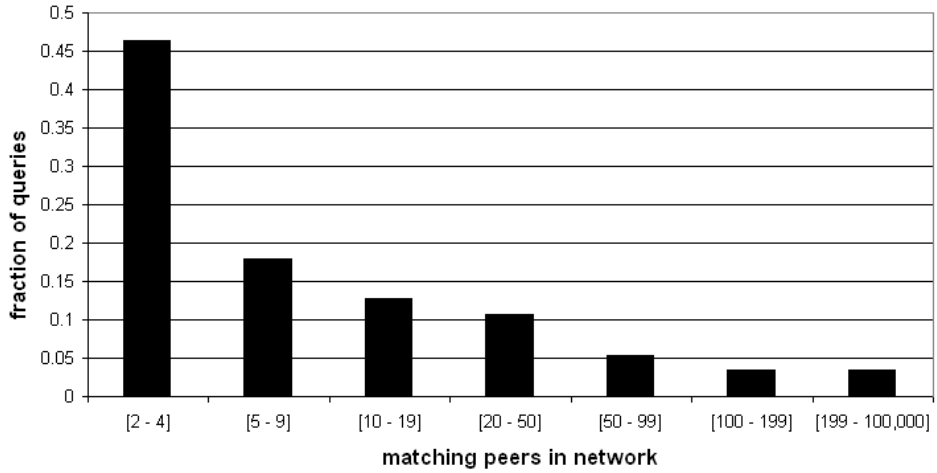

Figure 4.3: A typical distribution of the number of matching peers per query used in all simulations.

Description	Value
Minimum and maximum number of terms in query	1,6
Total number of queries per simulation experiment	20.000

Table 4.1: Static values for the parameters in the simulations

4.4.4 System parameters

In this subsection we describe the parameters that we implemented in our simulation platform. To keep the number of experiments within reasonable proportions, we unfortunately cannot do all permutations of set of values for the different parameters. To keep our document readable, we fix some of the parameters shown in Table 4.1. The default values which only vary during when we want to test their specific influence are shown in Table 4.2. The default and static values of these parameters are found by performing random experiments and choose those that gave good results.

4.5 Results

In this subsection, we show different sets of experiments in which we vary the parameters that had much influence on the results. For each set we show a figure that gives for 1 to 25 query rounds the average recall per round. Note

Description	Value
Total number of nodes in the system	100,000
Maximum number of expertise descriptions in a peer's register	5
Maximum number of expertise descriptions in a peer's cache	50
Maximum number of recommendations given by a register	5
Maximum number of recommendations given by a cache	30
Maximum number of advertisement rounds per advertisement initialization	5
Maximum number of neighbors selected per advertisement round	4
Maximum number of neighbors selected per query round	7
Average number of terms to register selected from expertise description	3

Table 4.2: Default values for the parameters in the simulations

that in a real setting the user can decide at which round he/she is satisfied and terminates the query process. All the results that we show are based on experiments which were performed three times, all using another part of the crawled data-set. This means that we get three independent sets of expertise-descriptions and queries, which reduces the chance that our results are very dependent on the data-set. Each individual result is the average of three identical simulation experiments using the three different data-sets. The standard-deviation for all results are smaller than 5%. One exception is the set of simulations that show the influence of different network sizes. We only did one data-set per experiment because for the experiments of 200K and 400K peers the data-set was too small to generate different partitions. Given the fact that the variance is small for all the other experiments, we can assume that this also will hold for these experiments.

Number of terms selected for registering. Figure 4.4 shows the influence of the number of terms that a peer selects from its expertise description for the registering and adverting process. When, for example, three terms are selected for registering, around 30% of the on average 32 peers relevant peers are found after 15 query rounds in the network of 100K peers (Fig. 4.3). This result of 30% is reached by using maximally $3_{(terms)} \times^2 \log(100.000)_{(DHT)} + 4_{(adv-neighbors)} \times 5_{(adv-rounds)} \approx 70$ messages for building the semantic overlay and $2.7_{(avg-query-length)} \times \log(100.000)_{(DHT)} + 7_{(query-neighbors)} \times 15_{(query-rounds)} \approx 150$ messages for the query process. When 25 terms are registered, a recall of 38% is reached but the number of advertisement messages is $25_{(terms)} \times^2 \log(100.000)_{(DHT)} + 4_{(adv-neighbors)} \times 5_{(adv-rounds)} \approx 435$, which is seven times more than the 70 messages from the previous example. This means that a relatively high price has to be paid to get this improvement of 8%. It has to be noted that the number of query messages stays the same. Besides this, it is important to look at the slope of the curves: the gain of

recall per round gets smaller after each round where more registered terms means a more steep curve than one with less registered terms. *Summarizing, these results show that with a fraction of the terms and a few advertisements, a good recall can be reached.*

Effect of varying the number of recommendations per query and storage capacity of a cache. Figure 4.5 shows the influence of the number of recommendations that a cache maximally returns and the maximum storage capacity. The results indicate that this capacity is much more important than the number of recommendations. For example, 10-50 (max. 10 recommendations per query and max. 50 descriptions stored per cache) gives better results than 40-40. The slope of the curves indicate that for the caches with a storage capacity of 30 and more, an small increase of the number of rounds (or query neighbors per round which is not shown in this paper) would result in an significant increase in recall. *The results confirm that it is no problem that there is a fixed and relative small cache size.*

Effect of varying the number of recommendations per query and storage capacity of a register. Figure 4.6 shows the influence of the number of recommendations that a register maximally returns and the maximum storage capacity of the register. In contrast to the previous results about the cache, the effect of both varying the number of recommendations and the storage capacity seems to be very small. First, please note that one of the characteristics of the data-set is that some terms occur more often in the different expertise descriptions than others. Therefore, the chance that such a term is selected as a key for the registering process is therefore higher than other terms, which means that registers on these popular terms receive more expertise descriptions to register. *Our results indicate that bounding the maximum to a small number (which is enough for most average registers), has only a small effect on the overall performance.*

The number of advertisements. Figures 4.7 and 4.8 show the influence of the number of advertisements that are send on average per peer per advertisement initialization. The graphs indicate that both increasing the number of neighbors to advertise to and the number of advertisement rounds have a positive effect on the recall. *The results confirm that only a relatively small number of advertisements are needed to build the semantic overlay.*

Effect of varying the generality parameter α. Figure 4.9 shows the effect of varying the α parameter in the generality formula 4.1 used by the register. The results indicate that from 1 to 15 rounds, where this parameter has more influence (i.e. specific terms gain more importance above generic terms), it has a positive effect on the recall. However due to the different

slopes of the lines, the lower values are better after approximately 20 rounds. Figure 4.10 indicates that varying α in the generality formula used by the cache seems to have opposite behavior as varying it at the register. Here, choosing a low value (e.g. 0.1) seems to be better if a quick high recall is desired but if a user has enough patience, it seems to be better to choose a value around 0.5. *Summarizing: the generality parameter in the register has influence on the results and the optimal value depends on the desired recall and patience of the user.*

Effect of varying the network size. Figure 4.11 shows the effect that different network sizes have on the performance of the system. In a network of 10.000 peers, only 70 messages are needed to build the register the terms and to build semantic overlay, to get a recall of more than 55% after 25 rounds. When we increase the number of peers 40 times, and keep the number of advertisement and query messages the same, the recall only is reduced to approximately 25%. Note that in a larger network there are also more matching peers which makes it more difficult for the protocol to find them all for a fixed number of query messages. *The result show that even for a large network of 400K peers, 10 query rounds are sufficient to find 20% of the relevant peers.*

4.6 Conclusion

Distributed Hash Tables (DHTs) and Semantic Overlay Networks (SONs) are two techniques to improve the efficiency of searching content in a Peer-to-Peer network. Both have their own drawbacks. DHTs have the disadvantage of (1) expensive maintenance costs in terms of the number of messages, (2) difficulty to deal with skewed distributions of content (load-balancing) and (3) vulnerability of single point of failures. SONs are cheaper in maintenance and load-balancing is an emergent property. One disadvantage of SONS is that most of them rely on a shared data-structure in which content has to be described. When this data-structure changes, the update has to be transmitted to all peers which results in much traffic. Another disadvantage is that peers which do not have any content of themselves cannot be clustered in the semantic overlay. In such cases, the assumption that a peer knows semantically related peers cannot be made.

In this paper we presented the pNear system that combines both techniques: it uses DHTs to add a register functionality to each peer, making them a kind of ‘yellow pages’ where peers can register expertise descriptions, which are sets of terms that describe the content that those peers share. The DHT approach allows a peer to efficiently find registers that are responsible for the terms in the query. Due to the clustering of expertise, the returned peers

by the register ‘know’ related peers by which the query has a good chance to be answered. In this way the rest of the potentially correct peers are found. By combining these two techniques we have the advantages of semantic overlays but allow also anonymous peers (peers that do not have of want to share content) to search efficiently in the network. Our simulation results indicate that the method finds a large fraction of the correct peers for a query originating from an unclustered peer. Also, only a small fraction of the terms from a peer’s expertise description need to be hashed and stored at the registers which reduces the DHT maintenance costs. Also only a small number of advertisement messages is needed to build the semantic overlay.

Some future work directions are to use more rich expertise descriptions and/or more advanced distance measures between those descriptions and between queries and expertise descriptions. For example, it would be interesting to use Latent Semantic Indexing to determine, via statistical analysis of text documents, the correlation between terms. In that way, the peers in the network can discover that ’dog’ and ’hound’ are related terms, and when this peer receives a query on term ’dog’ it could forward it to the peer that has the term ’hound’ in its expertise description.

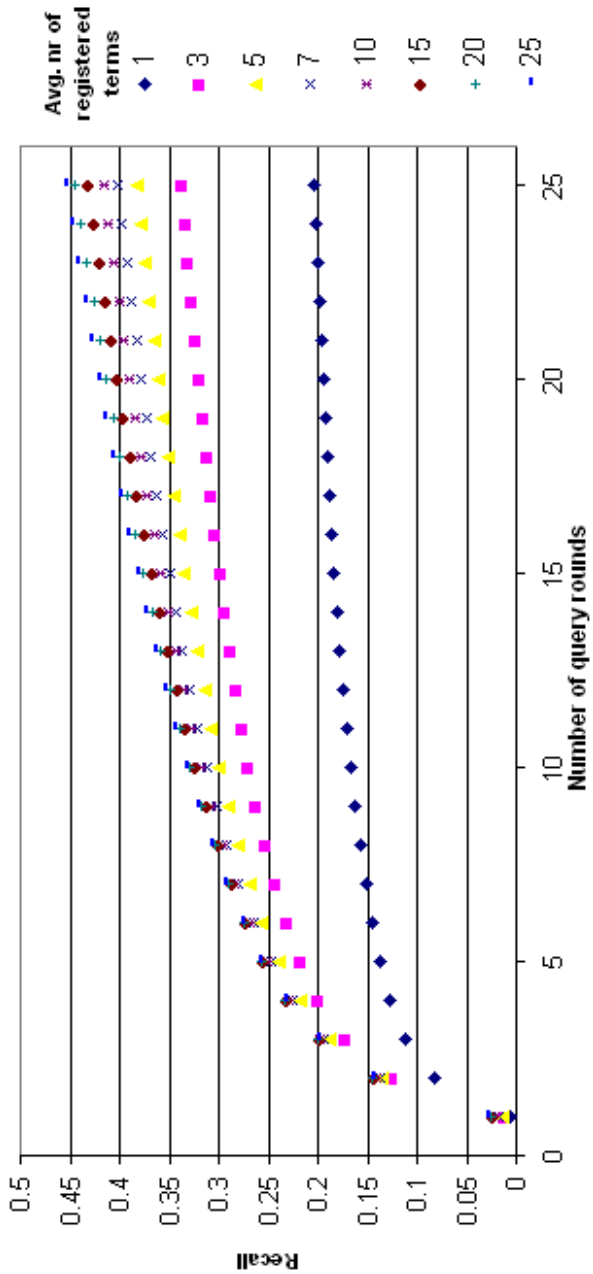

Figure 4.4: Influence of varying the average number of terms that are registered per peer.

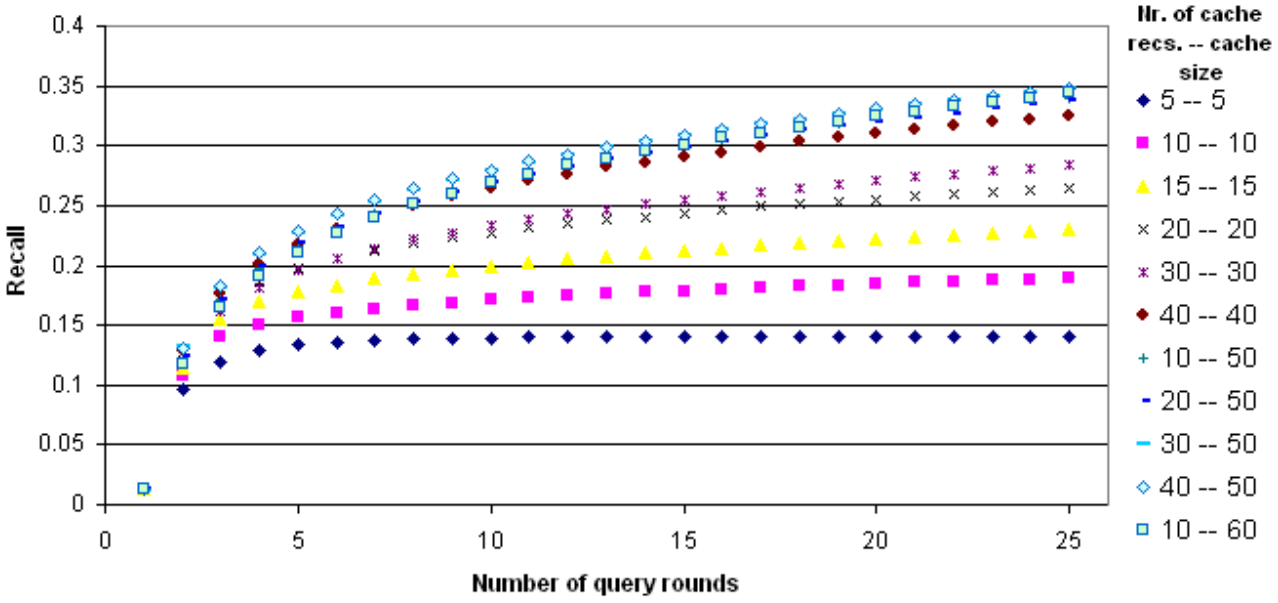

Figure 4.5: **Effect of varying the number of recommendations per query and storage capacity of a cache.** The notation in the legend should be read as follows: e.g. 20-50 means max. 20 recommendations and max. 50 expertise descriptions in a peer's cache.

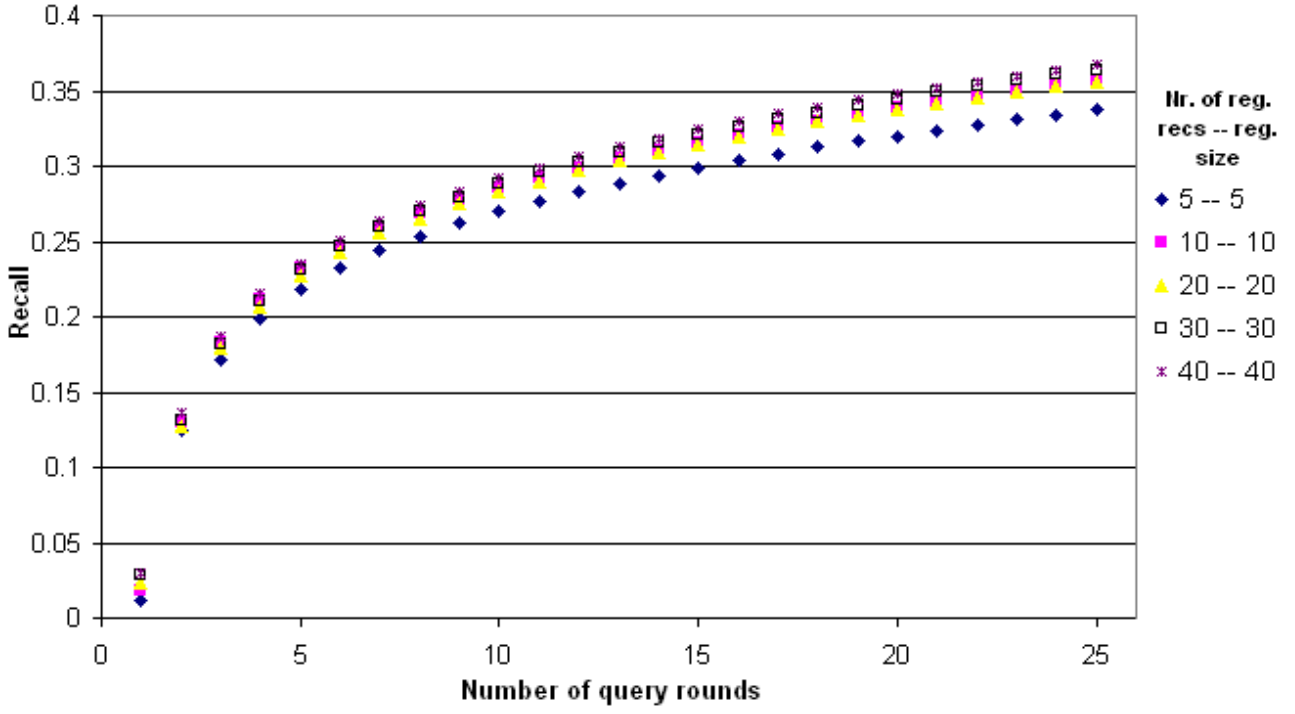

Figure 4.6: **Effect of varying the number of recommendations per query and storage capacity of a register.** The notation in the legend should be read as follows: e.g. 20-50 means max. 20 recommendations and max. 50 expertise descriptions in a peer's register.

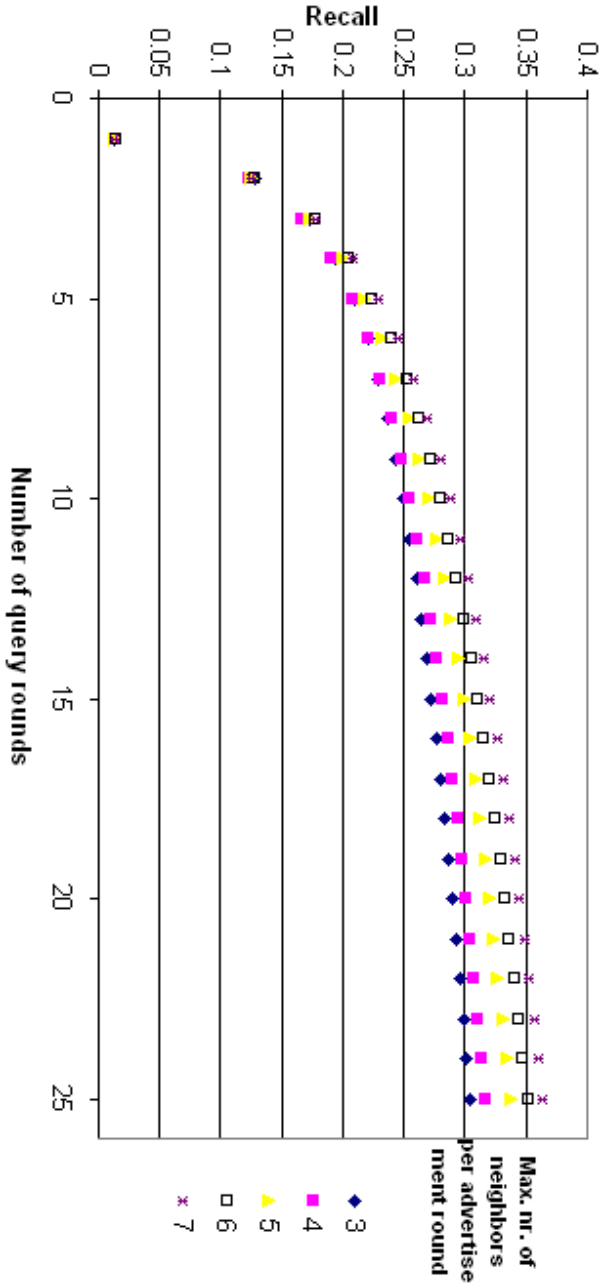

Figure 4.7: **Effect of varying the maximum number of neighbors selected per round in an advertisement process.**

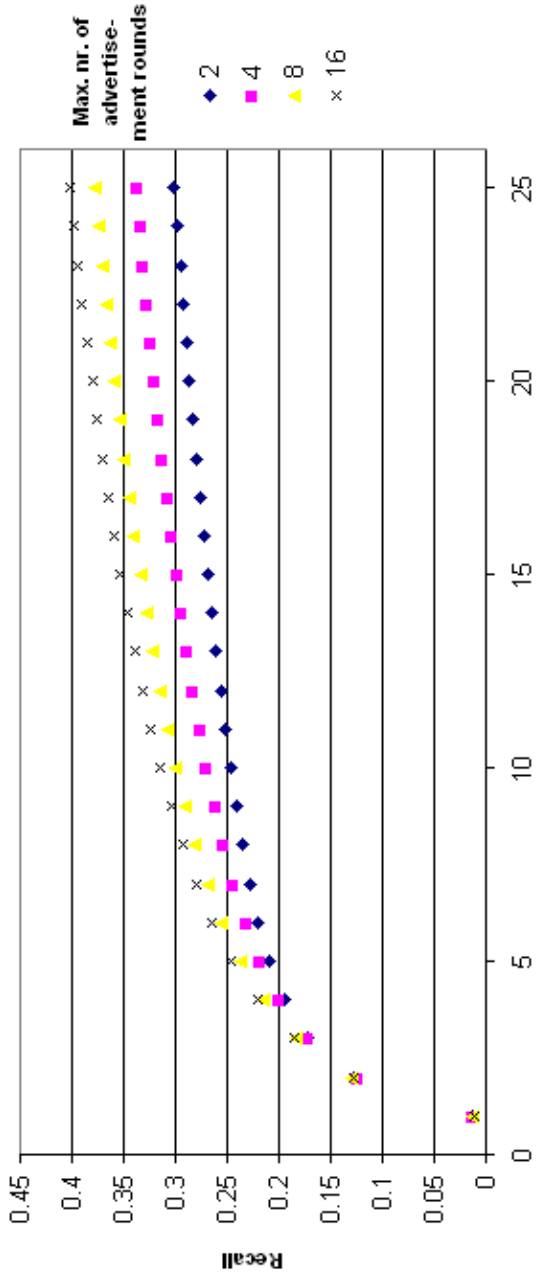

Figure 4.8: **Effect of varying the maximum number of advertisement rounds.**

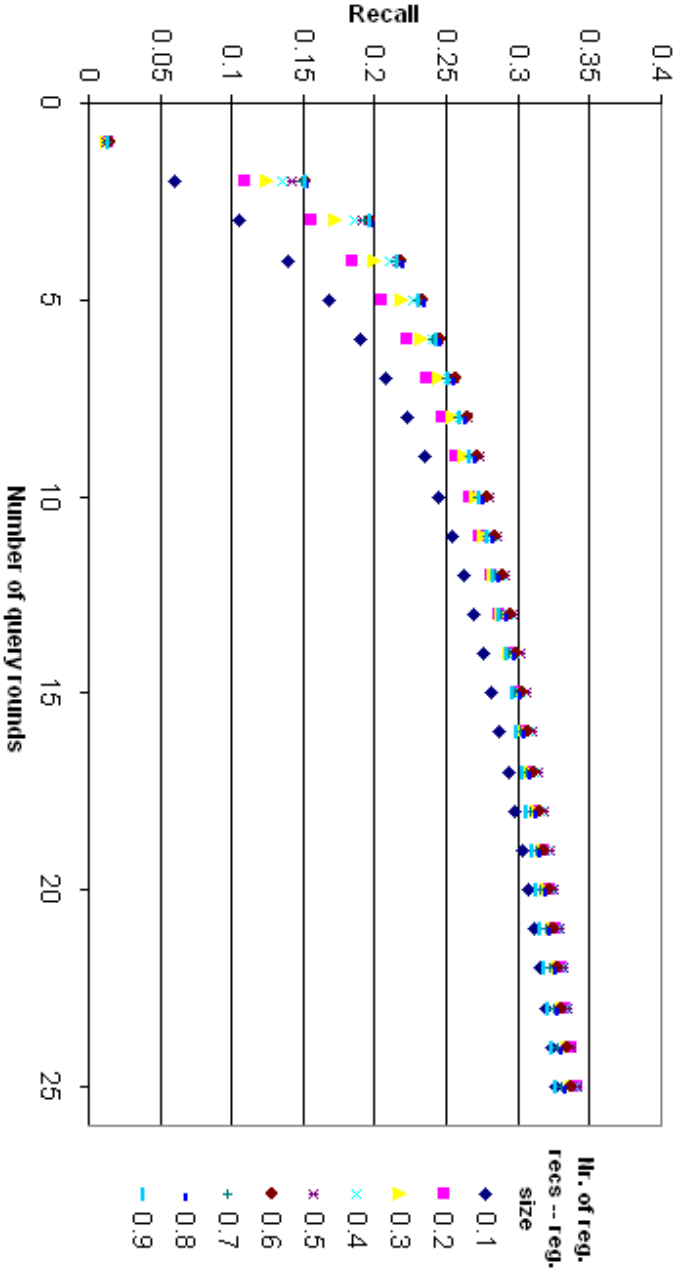

Figure 4.9: **Effect of varying the generality parameter in the register.**

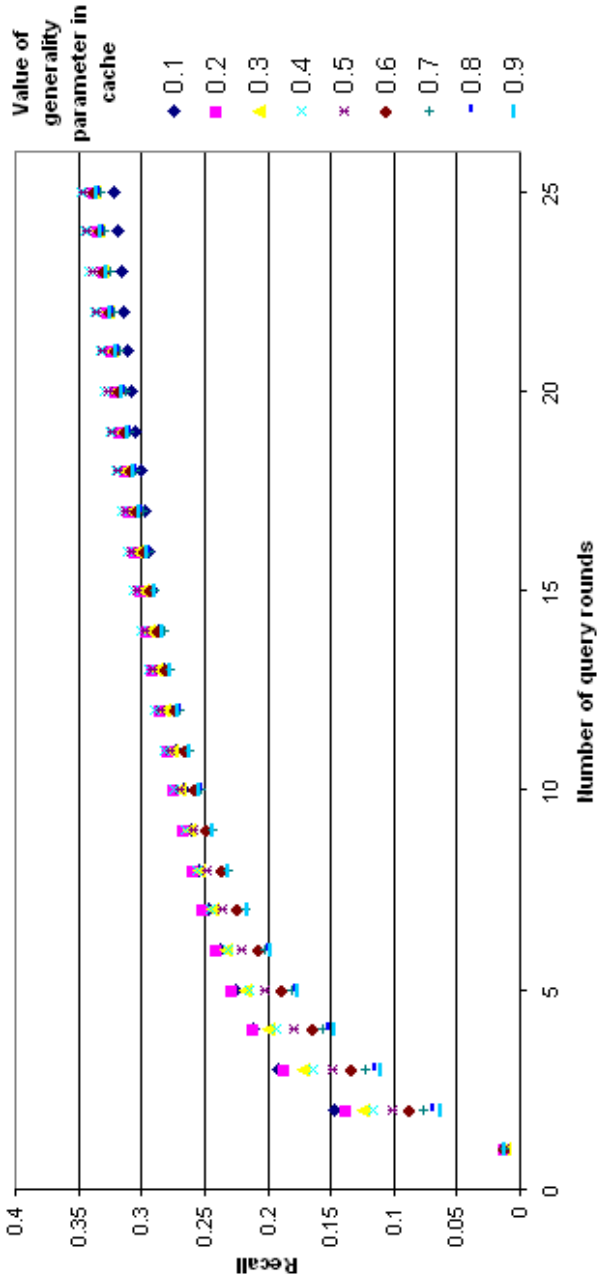

Figure 4.10: **Effect of varying the generality parameter in the cache.**

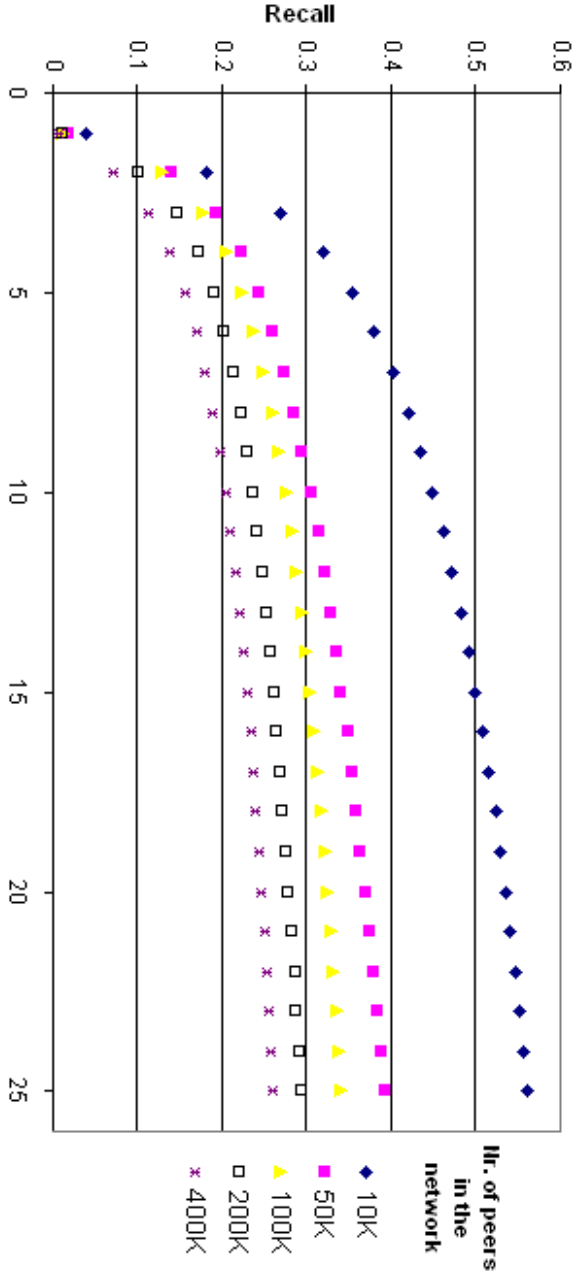

Figure 4.11: **Effect of varying the network size.**

Chapter 5

Conclusions and future work

This final chapter of the thesis summarizes the results of the three main chapters (Chapters 2, 3 and 4) and tries to look at future research directions. The goal of all the three approaches is identical, namely finding ways to efficiently route query messages to peers, which have relevant content to the queries, in a completely distributed P2P system. The three approaches all belong to the category of 'Semantic-Overlay-Networks' (SONs). In a SON, peers maintain pointers to semantically relevant peers based on content descriptions, which makes them able to choose the relevant peers for queries instead of, for example, choosing random peers. In each of the three chapters we compare the corresponding approach with relevant work and describe why the approach is unique and relevant and what the assumptions are. The following three sections summarizes the approaches and the conclusions of field- and simulation experiments.

5.1 Conclusions

5.1.1 Bibster

In chapter 2, Bibster is introduced as a model for using content descriptions of peers written in terms occurring in a shared ontology. The goal of these descriptions is to advertise them to some peers in the network, which makes them in their turn able to forward queries to only those peers that semantically closely match the content of the query (also described in terms occurring in the shared ontology). This matching is done by letting peers have a shared similarity metric on terms in the ontology. This metric allows

peers to calculate the semantic 'distance' between two sets of terms, being the terms in the expertise descriptions and in the queries. In the chapter we instantiate the model with a bibliographic scenario, where researchers share their bibliographical entries on a P2P network. The Bibster method is evaluated by simulating the bibliographic scenario on a realistic data-set and by performing a field study by examining log-files of the Bibster application. The conclusions:

- Using expertise-based peer selection with a topic-hierarchy as instantiation of the shared data-structure, in a bibliographic scenario, can increase the performance of the peer selection by an order of magnitude compared with a random selection algorithm
- However, if expertise-based peer selection uses simple exact matching via set-overlap between expertise terms and query terms, the recall drops to unacceptable levels. It is necessary to use an ontology-based similarity measure as the basis for expertise-based matching.
- An advertising strategy where peers only accept advertisements which are semantically close to their own profile (i.e. that are in their semantic neighborhood) is a simple and effective way of creating a semantic topology which increases the performance compared to random acceptance. This semantic topology allows to forward queries along the gradient of increasing semantic similarity.
- The above results depend on how closely the semantic topology of the network mirrors the structure of the ontology. All relevant performance measures reach their optimal value when the network is organized exactly according to the structure of the ontology.

5.1.2 pRoute

In chapter 3, pRoute replaced the manually hand-crafted ontology from the previous approach by a term similarity matrix. Please recall that this ontology was used by peers in a Bibster system to express their expertise and to determine semantic similarity between queries and expertise descriptions. The term-similarity matrix explicitly stated the semantic distances between the corresponding terms. Such a matrix can be extracted manually (for example by applying the semantic distance metric on the ontologies in Bibster), but also automatically via, for example, the LSI-method. Via much more and larger simulations than in the Bibster paper, we showed that we can get comparable results to the Bibster approach with the advantage that the shared datastructure is automatically generated. The main conclusions:

- A small random subset of a large set of documents in the computer science domain is sufficient to generate automatically a representative similarity matrix for the whole domain. More precisely, 10% of 100K

documents is enough to get 90% of the relevant terms from all the documents.

- A term similarity matrix constructed by the LSI algorithm is a good candidate for routing messages to relevant peers because it gives a very good precision and recall for queries in the network.
- Simulation results indicate that a pRoute system is scalable in network size and robust to frequent joins and leaves of peers
- Even with low availability, the pRoute algorithm outperforms the random counterpart in terms of precision, recall and bandwidth usage.
- The results show that the performance of the system is sensitive to parameters and also sometimes a bit counter intuitive. For example increasing the cache size is only benificial until a maximum, after which the performance decreases again. The network size and availability of peers dictate the optimal parameter settings.

5.1.3 pNear

pNear is the last routing algorithm of this thesis. Instead of letting peers have shared data-structures for describing their content, it combines two existing techniques: Distributed Hash Tables (DHT) and content clustering. The goal of this combination is to combine the advantages of both techniques without inheriting the disadvantages that hold for each unique approach. The reason for having this routing protocol besides the other two described in this thesis, is that in pNear peers do not have to agree on shared data-structures. This could happen in situations where information is very versatile and/or change very frequently or where it is impossible to classify it into the data-structure. The way pNear works is to let some peers register their expertise at 'yellow-page' peers, i.e. peers that maintain lists of peers that have registered themselves for that expertise. The clustering technique allows peers to form clusters of expertise, which means that the neighbors of a clustered peer have similar expertise. Now the yellow page peer can be seen as a starting point for a query, where the register returns one or more pointers to a relevant peer. And then these peers can, besides perhaps answering the query, forward the query to their neighbors (which are relevant due to the clustering of expertise). The main conclusions:

- We gathered a rich and realistic data-set which we think can be valuable for researchers that want to do simulations.
- pNear gives very good results in terms of recall and low network usage with only registering a fraction of the terms that would be needed in a classical DHT approach
- pNear is scalable in terms of performance to large networks.

- The performance can be reached by very few advertisement messages
- The performance can be reached by very small advertisement caches and register caches.

5.2 Outlook

In this section we try to come up with some interesting future work building on the results from this thesis.

5.2.1 Concluding comparison between the three approaches

Although we already mentioned the relaxations of the assumptions from chapter to chapter and also compared the results of the various experiments, it would be good to have a concluding comparison of all approaches in exactly the same setting, by which we mean the data-set, user queries and availability.

5.2.2 Extending current experiments

Given that writing the simulation platforms and gathering the large data-sets are time-consuming tasks, an easy and efficient thing to do is relaxing various key assumptions made in the three sections by implementing extra functionality on the existing platforms. Some interesting assumptions to drop are:

- **Shared similarity measure.** During each simulation and field experiment peers all have the same way to calculate the semantic relatedness between queries and expertise descriptions. Due to the complete autonomy of all peers, this assumption is not needed and is only made to limit the amount of implementation work. It would be interesting to see what the influence on the performance would be when peers can differ in their way of calculating this similarity.
- **Shared data-structures.** In Bibster and pRoute, it is very likely that peers do not all need to have the same identical data-structure (the topic-hierarchy in Bibster, and the similarity matrix in pRoute) to get a good performance. Probably there is a correlation between performance and the partial overlap of data-structures. Future research could try to find the functional dependency, for example if there is a linear dependency.

- **Stable network.** In the Bibster and pNear experiments we did not alter the availability of peers in the network. Therefore, we do not now how robust the system is to frequent node drops/joins. Clearly, this is an important thing to know and therefore is at the top on my personal wish list.

5.2.3 Remove key limitations of current approaches

Decisions made in the design process of the different systems have also lead to serious limitations. For example, pRoute is not able to deal with very low availability. It has to be said however that many other approaches described in literature are not tested with low availability or also have bad performance. One solution to solve this performance problem could be to let peers be aware of the average availability. When this average is low, peers could advertise more to become more visible in the network. A limitation of Bibster is its simple advertising protocol, namely one advertisement round which results in a fixed topology. In pRoute and pNear this was different. It would be fair, when we compare the different approaches, to adjust the Bibster approach to have a more dynamic advertisement strategy.

5.2.4 Moving results out of the lab

In the SWAP project we contributed to the implementation of the Bibster system which was actually quite a success given the number of downloads, the publicity and the two software awards. Due to time constraints, pNear and pRoute remained only to be tested via simulations. Namely, there are still some open questions which need to be answered before we can be sure that everything works as hoped. First, in pRoute we simulated human behavior by generating queries and peer joins and leaves. In pNear we made a feasible claim that the algorithm is robust because it inherits the properties of a semantic overlay network, however it would much better to strengthen the claim by either simulation experiments or a real field-study. In the OpenKnowledge project which is like SWAP also funded by the EU, we try to build a real system based on the experiences of this thesis.

Other future work lies in moving the case-studies and or data-sets to other domains. For now, the corpora in Bibster and pRoute remained within the computer-science domain. It would be interesting to see how it behaves in domains where, for example, the distribution of data has different characteristics. The results of pNear suffered less from this distribution bias because they are based on a large, realistic en generic data-set.

5.2.5 New possibilities

The last type of future work lies in new possibilities triggered by the results of this thesis. Firstly, in all the three routing proposals, there is no reputation management. This means that when a peer A sends its expertise description to a peer B, peer B believes peer A. In our simulations there was not a reason to distrust peers, because we made them this way. However, in the real world it could be beneficial to lie about expertise, for example to attract more attention or to spread malicious rumors. Secondly, in our approaches peers are not aware of their local position in the network- reputation management. When peers are more aware about their position, they are more able to set priorities between different tasks. The research area on Social Networks, can provide more insight into this matter and therefore it would be interesting to combine efforts. Another aspect of this research area is that they also investigate the properties of large dynamic networks. Given that a our P2P networks are also a kind of social networks, it is very likely that analyzing the global properties could lead to more efficient adjusting of the parameters to increase the network performance.

Bibliography

[acm,] The ACM Topic Hierarchy.
`http://www.acm.org/class/1998/`.

[Aberer, 2001] Aberer, K. (2001). P-grid: A self-organizing access structure for p2p information systems. In *Sixth International Conference on Cooperative Information Systems (CoopIS 2001),*, volume 2172 of *Lecture Notes in Computer Science*, Trento, Italy. Springer Verlag.

[Aberer et al., 2003] Aberer, K., Cudré-Mauroux, P., Datta, A., Despotovic, Z., Hauswirth, M., Punceva, M., and Schmidt, R. (2003). P-grid: a self-organizing structured p2p system. *SIGMOD Rec.*, 32(3):29–33.

[Aberer et al., 2004] Aberer, K., Cudré-Mauroux, P., Hauswirth, M., and Pelt, T. V. (2004). Gridvine: Building internet-scale semantic overlay networks. In [McIlraith et al., 2004], pages 107–121.

[Ahlborn et al., 2002] Ahlborn, B., Nejdl, W., and Siberski, W. (2002). OAI-P2P: A peer-to-peer network for open archives. In *2002 International Conference on Parallel Processing Workshops (ICPPW'02)*.

[Bayardo, Jr. et al., 1997] Bayardo, Jr., R. J., Bohrer, W., Brice, R., Cichocki, A., Fowler, J., Helal, A., Kashyap, V., Ksiezyk, T., Martin, G., Nodine, M., Rusinkiewicz, M., Shea, R., Unnikrishnan, C., Unruh, A., and Woelk, D. (1997). InfoSleuth: Agent-based semantic integration of information in open and dynamic environments. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, volume 26,2, pages 195–206, New York. ACM Press.

[Berners-Lee et al., 2001] Berners-Lee, T., Hendler, J., and Lassila, O. (2001). The semantic web. *Scientific American*.

[Berry et al., 1999] Berry, M. W., Drmac, Z., and Jessup, E. R. (1999). Matrices, vector spaces, and information retrieval. In *SIAM Review*, pages 335–362.

[Breslau et al., 1999] Breslau, L., Cao, P., Fan, L., Phillips, G., and Shenker, S. (1999). Web caching and zipf-like distributions: Evidence and implications. In *Proceedings of the IEEE INFOCOM'99 conference*, pages 126–134, New York, USA.

[Broekstra et al., 2004] Broekstra, J., Ehrig, M., Haase, P., van Harmelen, F., Menken, M., Mika, P., Schnizler, B., and Siebes, R. (2004). Bibster - a semantics-based bibliographic peer-to-peer system. In *Proceedings of the WWW'04 Workshop on Semantics in Peer-to-Peer and Grid Computing, New York, 2004*.

[Broekstra and Kampman, 2004] Broekstra, J. and Kampman, A. (2004). SeRQL: An RDF Query and Transformation Language. Submitted to the International Semantic Web Conference, ISWC 2004. See also http://www.openrdf.org/doc/SeRQLmanual.html.

[Byers et al., 2002] Byers, J., Considine, J., and Mitzenmacher, M. (2002). Simple load balancing for distributed hash tables. Technical report, CS Department, Boston University.

[Clarke et al., 2001] Clarke, I., Sandberg, O., Wiley, B., and Hong, T. (2001). Freenet: A distributed anonymous information storage and retrieval system. In *Proc. Int'l Workshop on Design Issues in Anonymity and Unobservability*, pages 46–66.

[Cohen et al., 2003] Cohen, E., Fiat, A., and Kaplan, H. (2003). Associative search in peer to peer networks: Harnessing latent semantics. In *Proceedings of the IEEE INFOCOM conference*, San Fransisco, CA, USA.

[Crespo and Garcia-Molina, 2002] Crespo, A. and Garcia-Molina, H. (2002). Routing indices for peer-to-peer systems. In *ICDCS '02: Proceedings of the 22 nd International Conference on Distributed Computing Systems (ICDCS'02)*, page 23, Washington, DC, USA. IEEE Computer Society.

[Cunningham et al., 1997] Cunningham, H., Humphreys, K., Gaizauskas, R., and Wilks, Y. (1997). Software infrastructure for natural language processing. In *5th Conference on Applied Natural Language Processing*, Washington.

[Cunningham et al., 1996] Cunningham, H., Wilks, Y., and Gaizauskas, R. (1996). New methods, current trends and software infrastructure for nlp. In *Proceedings of the Second Conference on New Methods in Language Processing*, pages 283–298, Ankara, Turkey.

[Deerwester et al., 1990] Deerwester, S. C., Dumais, S. T., Landauer, T. K., Furnas, G. W., and Harshman, R. A. (1990). Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407.

[Ehrig et al., 2003a] Ehrig, M., Schmitz, C., Staab, S., Tane, J., and Tempich, C. (2003a). Towards evaluation of peer-to-peer-based distributed knowledge management systems. In *Proceedings of the AAAI Spring Symposium "Agent-Mediated Knowledge Management (AMKM-2003)"*.

[Ehrig et al., 2003b] Ehrig, M., Tempich, C., Broekstra, J., van Harmelen, F., Sabou, M., Siebes, R., Staab, S., and Stuckenschmidt, H. (2003b).

A metadata model for semantics-based peer-to-peer systems. In *Proceedings of the second Konferenz Professionelles Wissensmanagement*, Lucern.

[Ehrig et al., 2003c] Ehrig, M., Tempich, C., Broekstra, J., van Harmelen, F., Sabou, M., Siebes, R., Staab, S., and Stuckenschmidt, H. (2003c). SWAP - ontology-based knowledge management with peer-to-peer technology. In Sure, Y. and Schnurr, H.-P., editors, *Proceedings of the 1st National "Workshop Ontologie-basiertes Wissensmanagement (WOW2003)"*, Luzern, Switzerland.

[Finin et al., 1994] Finin, T., Fritzson, R., McKay, D., and McEntire, R. (1994). KQML as an Agent Communication Language. In Adam, N., Bhargava, B., and Yesha, Y., editors, *Proceedings of the 3rd International Conference on Information and Knowledge Management (CIKM'94)*, pages 456–463, Gaithersburg, MD, USA. ACM Press.

[Gruber, 1993] Gruber, T. R. (1993). Towards Principles for the Design of Ontologies Used for Knowledge Sharing. In Guarino, N. and Poli, R., editors, *Formal Ontology in Conceptual Analysis and Knowledge Representation*, Deventer, The Netherlands. Kluwer Academic Publishers.

[Haase et al., 2004a] Haase, P., Broekstra, J., Ehrig, M., Menken, M., Mika, P., Olko, M., Plechawski, M., Pyszlak, P., Schnizler, B., Siebes, R., Staab, S., and Tempich, C. (2004a). Bibster - a semantics-based bibliographic peer-to-peer system. In [McIlraith et al., 2004], pages 122–136.

[Haase et al., 2004b] Haase, P., Broekstra, J., Ehrig, M., Menken, M., Mika, P., Plechawski, M., Pyszlak, P., Schnizler, B., Siebes, R., Staab, S., and Tempich, C. (2004b). Bibster - a semantics-based bibliographic peer-to-peer system. In *Proceedings of the Third International Semantic Web Conference, Hiroshima, Japan, 2004*.

[Haase et al., 2004c] Haase, P., Siebes, R., and van Harmelen, F. (2004c). Peer selection in peer-to-peer networks with semantic topologies. In Bouzeghoub, M., editor, *Proceedings of the International Conference on Semantics in a Networked World (ICNSW'04)*, volume 3226 of *LNCS*, pages 108–125, Paris. Springer Verlag.

[Kan, 2001] Kan, G. (2001). Gnutella. In Oram, A., editor, *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*, pages 94–122. O'Reilly and Associates.

[Lassila and Swick, 1999] Lassila, O. and Swick, R. R. (1999). Resource description framework (rdf) model and syntax specification. `http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/`.

[Leibowitz et al., 2003] Leibowitz, N., Ripeanu, M., and Wierzbicki, A. (2003). Deconstructing the kazaa network. 3rd IEEE Workshop on Internet Applications (WIAPP'03). Santa Clara, CA.

[Li et al., 2003] Li, Y., Bandar, Z. A., and McLean, D. (2003). An approach for measuring semantic similarity between words using multiple information sources. *Transactions on Knowledge and Data Engineering*, 15(4):871–882.

[Löser et al., 2003] Löser, A., Wolpers, M., Siberski, W., and Nejdl, W. (2003). Efficient data store discovery in a scientific P2P network. In Ashish, N. and Goble, C., editors, *Proceedings of the WS on Semantic Web Technologies for Searching and Retrieving Scientific Data*, CEUR WS 83. Colocated with the 2. ISWC-03.

[Maedche and Staab, 2001] Maedche, A. and Staab, S. (2001). Ontology learning for the semantic web. *IEEE Intelligent Systems*, 16(2):72–79.

[McIlraith et al., 2001] McIlraith, S., Son, T., and Zeng, H. (2001). Semantic web services. In *IEEE Intelligent Systems: Special Issue on the Semantic Web*.

[McIlraith et al., 2004] McIlraith, S. A., Plexousakis, D., and van Harmelen, F., editors (2004). *The Semantic Web - ISWC 2004: Third International Semantic Web Conference,Hiroshima, Japan, November 7-11, 2004. Proceedings*, volume 3298 of *Lecture Notes in Computer Science*. Springer.

[Nejdl et al., 2002] Nejdl, W., Wolf, B., Qu, C., Decker, S., Sintek, M., Naeve, A., Nilsson, M., Palmer, M., and Risch, T. (2002). Edutella: A p2p networking infrastructure based on rdf. In *Proceedings of the 11th International World Wide Web Conference*. schema based searching Presentation: http://www2002.org/presentations/nejdl.pdf.

[Nejdl et al., 2003] Nejdl, W., Wolpers, M., Siberski, W., Schmitz, C., Schlosser, M., Brunkhorst, I., and Löser, A. (2003). Super-peer-based routing and clustering strategies for rdf-based peer-to-peer networks. In *Proceedings of the 12th International World Wide Web Conference, Budapest, Hungary, May 2003.*

[Rada et al., 1989] Rada, R., Mili, H., Bicknell, E., and Blettner, M. (1989). Development and application of a metric on semantic nets. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(1):17–30.

[Ratnasamy et al., 2001] Ratnasamy, S., Francis, P., Handley, M., Karp, R., and Shenker, S. (2001). A scalable content-addressable network. In *Proceedings of ACM SIGCOMM '01*.

[Rivest, 1992] Rivest, R. (1992). Rfc 1321: The md5 message-digest algorithm. Internet Activities Board.

[Rowstron and Druschel, 2001] Rowstron, A. and Druschel, P. (2001). Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, pages 329–350, Heidelberg, Germany.

[Schlosser et al., 2002] Schlosser, M. T., Sintek, M., Decker, S., and Nejdl, W. (2002). Hypercup - hypercubes, ontologies, and efficient search on peer-to-peer networks. In Moro, G. and Koubarakis, M., editors, *AP2PC*, volume 2530 of *Lecture Notes in Computer Science*, pages 112–124. Springer.

[Siebes,] Siebes, R. pnear - combining content clustering and distributed hash tables. Journal on Data Semantics (Springer LNCS), year = 2006, owner = ronny,.

[Siebes, 2005] Siebes, R. (2005). pnear - combining content clustering and distributed hash tables. In *Proceedings of the IEEE'05 Workshop on Peer-to-Peer Knowledge Management.*, San Diego, CA, USA.

[Siebes et al., 2006] Siebes, R., Haase, P., and van Harmelen, F. (2006). Expertise-based peer selection in peer-to-peer networks. To appear in: Journal of Knowledge and Information Systems.

[Siebes and Kotoulas, 2005] Siebes, R. and Kotoulas, S. (2005). proute: Expertise-based selection using shared term similarity matrices. In Verbeeck, K., Tuyls, K., Nowé, A., Manderick, B., and Kuijpers, B., editors, *Proceedings of the 17th Belgian-Dutch Conference on Artificial Intelligence*, pages 202–208, Brussels, Belgium. Contactforum.

[Siebes and Kotoulas, 2006] Siebes, R. and Kotoulas, S. (2006). proute: Peer selection using shared term similarity matrices. To appear in: Journal of Web Intelligence and Agent Systems.

[Sripanidkulchai et al., 2003] Sripanidkulchai, K., Maggs, B., and Zhang, H. (2003). Efficient content location using interest-based locality in peer-to-peer systems. In *Proceedings of the IEEE INFOCOM conference*, San Fransisco, CA, USA.

[Stoica et al., 2001] Stoica, I., Morris, R., Karger, D., Kaashoek, M. F., and Balakrishnan, H. (2001). Chord: A scalable peer-to-peer lookup service for Internet applications. In *Proceedings of the ACM SIGCOMM '01*.

[Tang et al., 2002] Tang, C., Xu, Z., and Dwarkadas, S. (2002). Peer-to-peer information retrieval using self-organizing semantic overlay networks. Technical report, HP Labs.

[Tempich et al., 2004] Tempich, C., Staab, S., and Wranik, A. (2004). Remindin': semantic query routing in peer-to-peer networks based on social metaphors. In *WWW '04: Proceedings of the 13th international conference on World Wide Web*, pages 640–649. ACM Press.

[Voulgaris et al., 2004] Voulgaris, S., Kermarrec, A.-M., Massoulie, L., and van Steen, M. (2004). Exploiting semantic proximity in peer-to-peer content searching. In *10th International Workshop on Future Trends in Distributed Computing Systems (FTDCS)*, Suzhou, China.

Samenvatting: Semantiek gebaseerde route selectie in Peer-to-Peer computer netwerken

Het vinden van informatie wordt vaak ondersteund door informatie technologie. Denk bijvoorbeeld aan een zoeksysteem in een bibliotheek of een systeem op het Internet zoals Google. Vaak zijn die systemen gecentraliseerd, wat betekent dat alle zoekopdrachten worden verwerkt door één (groot) computersysteem. Zo'n systeem bevat meestal óf een handmatig ingevoerde index, zoals bij veel bibliotheken óf wordt informatie automatisch geïndexeerd, zoals bij Google. Het grote voordeel van een gecentraliseerd zoeksysteem is dat er weinig netwerkverkeer nodig is om de vragen beantwoord te krijgen, namelijk maar één bericht voor de vraag en meestal een paar berichten voor de antwoorden. Als de computer, die de vragen moet beantwoorden, snel is (zoals bij Google), worden binnen enkele miliseconden de resultaten aan de gebruiker getoond. In veel situaties werkt deze gecentraliseerde aanpak dus ook prima, maar er zijn ook enkele nadelen te noemen. Een recent voorbeeld is het faciliteren van censuur door twee grote Internet zoekmachines voor een aziatisch land. Op deze manier kon en kan de desbetreffende regering invloed uitoefenen op welke resultaten van de zoekmachine aan de gebruikers worden getoond. Ook wanneer er geen censuur wordt gepleegd, heeft de zoekmachine de volledige controle over welke informatie en in welke volgorde deze informatie wordt getoond. Bedrijven zouden vervolgens de eigenaren van de zoekmachine kunnen betalen om hun resultaten boven de resulaten van de concurrent te tonen. Het spreekt voor zich dat deze resultaten niet altijd ook de beste hoeven te zijn in de ogen van de gebruikers. Een ander nadeel van een gecentraliseerd zoeksysteem is de eenvoudige mogelijkheid tot het schenden van privacy. Gezien iedere computer een uniek adres heeft waarvan informatie wordt ontvangen en verstuurd, is het eenvoudig om een profiel op te maken van de zoekopdrachten

die komen van dat adres. Vervolgens is het gemakkelijk om automatisch dit profiel te vergelijken met een profiel dat bijvoorbeeld als 'staatsgevaarlijk' wordt aangemerkt en kan de gebruiker wellicht een bezoekje van de veiligheidsdienst verwachten. Hoewel potentiële terroristen misschien vroegtijdig kunnen worden ontdekt, ben ik van overtuiging dat het afluisteren van, in beginsel, onschuldige burgers hier niet tegenop weegt.

De twee genoemde nadelen zijn genoeg redenen om naar het voor de hand liggende alternatief te kijken, namelijk gedecentralizeerde zoeksystemen. Een zuiver gedecentraliseerd zoeksysteem bestaat uit een netwerk van aan elkaar gekoppelde computers waarbij ieder lid dezelfde functionaliteiten heeft, wat betekent dat er geen hiërarchische organisatie structuur bestaat. In zulke systemen worden zoekopdrachten die zijn geïnitieerd ergens op een computer in het netwerk, verstuurd naar enkele andere computers die zich in hetzelfde netwerk bevinden en wellicht de vraag van de initiator kunnen beantwoorden. Meestal stopt het zoekproces wanneer het maximum aantal berichten is bereikt of wanneer er voldoende antwoorden zijn. Door het ontbreken van hiërarchie in de organisatiestructuur van het netwerk en de daardoor onvoorspelbare route van de zoekopdrachten, is het per definitie een stuk lastiger om censuur te plegen omdat iedere computer slechts invloed heeft op zijn eigen ontvangen zoekopdrachten en niet op die die door anderen worden verwerkt. Ook is het een stuk lastiger om een profiel van een gebruiker op te maken omdat slechts een fractie of misschien wel geen enkele zoekopdracht van een bepaalde gebruiker aankomen bij een andere gebruiker. Voordat ik verder in ga op zuiver gedecentraliseerde zoeksystestemen, behandel ik eerst nog even de semi-gedecentraliseerde oplossingen waarbij sommige onderdelen gecentralizeerd zijn en sommige niet. Het succes van semi-gedecentralizeerde systemen is inmiddels al aangetoond gezien de populariteit van de fileshare-systemen als Napster en KaZaa[1], instant messaging systemen zoals MSN-messenger. Deze Peer-to-Peer (P2P) systemen hebben de prettige eigenschap dat de lasten in termen van bandbreedte en opslagruimte worden verdeeld over de leden in het netwerk. Dit resulteert dus in een enorm voordeel ten opzichte van een compleet gecentraliseerde oplossing. Hoewel het versturen van data (wat veel bandbreedte vergt) direct van aanbieder naar vrager wordt verstuurd, wordt het vinden van de aanbieder op de eerste plaats nog steeds vaak gedaan door een gecentralizeerd systeem. Met andere woorden, bij deze semi-gedecentraliseerde systemen gelden nog steeds maar in iets mindere mate de twee nadelen van mogelijke censuur en privacy schending die eerder genoemd zijn bij gecentralizeerde oplossingen. De reden voor het kiezen van een gecentralizeerde aanpak voor het zoeken van de desbetreffende systemen is dat wanneer informatie eenmaal geïndexeerd is, het

[1]KaZaa maakt gebruikt van een zgn. 'super-node' architectuur, waarbij krachtige systemen in het netwerk zich aan kunnen melden als zoekmachine die daardoor verantwoordelijk worden voor een groot aantal zoekopdrachten binnen het netwerk. Hierdoor is ook KaZaa een semi-gedecentraliseerde aanpak zodat nog steeds een kleine groep krachtige computers de mogelijkheid hebben om censuur te plegen of om verkeer af te luisteren

beantwoorden van een vraag niet veel computer kracht vergt en daardoor gemakkelijk door één systeem kan worden uitgevoerd en daardoor ook de volledige controle kan blijven houden over hoe en welke resultaten worden gevonden. Ook is het voordeel dat er heel weinig berichten nodig zijn om de aanbieder en de vrager te koppelen. Dit is veel lastiger als het zoeksysteem zelf ook gedecentraliseerd is zoals in Gnutella. Over precies dit onderwerp gaat dit proefschift:

"Hoe kunnen we in een gedecentraliseerd zoeksysteem het aantal berichten die nodig zijn om aanbieders te vinden voor een zoekopdracht verkleinen zodat een acceptabel alternatief ontstaat voor een (semi-)gecentraliseerd zoeksysteem?"

Dit proefschrift is een bundeling van drie artikelen die elk een eigen aanpak bespreken om het aantal berichten te reduceren voor het gewenste percentage correcte antwoorden. In alle artikelen dient het 'expertise-based selection' model als basis voor de, in de artikelen voorgestelde, gedistribueerde zoek algoritmen. Het model is geschetst in figuur 5.1 en uitgebreid beschreven in [Haase et al., 2004c].

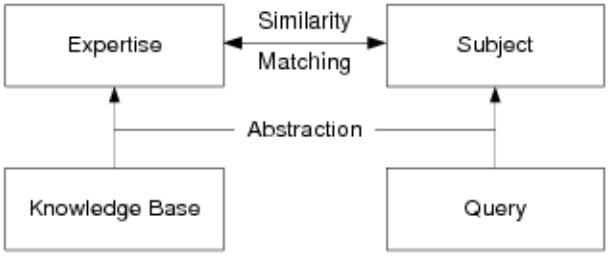

Figure 5.1: Expertise Based Matching

In dit model beschrijven de computers de informatie die ze aanbieden op het netwerk in zogenoemde *expertise descriptions* (expertise beschrijvingen) en zoekopdrachten in *query abstractions* (zoekopdracht abstracties). De expertise beschrijvingen zijn samenvattingen en/of abstracties van de informatie die aangeboden wordt. Zo zou het kunnen zijn dat wanneer een computer tekstdocumenten aanbiedt, de meest interessante termen uit die documenten in de expertise beschrijvingen staan. Vervolgens worden deze verspreid via "advertentie berichten" naar andere computers binnen het netwerk. Deze kennis wordt door computers gebruikt om voor binnenkomende zoekopdrachten, die ze zelf ook proberen te beantwoorden, dié kandidaten te selecteren waarvan de inhoud van de advertentie berichten het beste aansluit bij de abstractie van de binnenkomende zoekopdracht.

Netwerken waar computers beschrijvingen de data van andere computers onthouden worden ook wel 'Semantic Overlay Networks' (SONs) genoemd. SONs maken het mogelijk om op een efficiente manier te zoeken in een P2P netwerk. In dit soort netwerken gebruiken computers namelijk hun kennis over de inhoud van andere computers om zo gericht zoekopdrachten door te sturen in plaats van willekeurig waardoor er veel gerichter wordt gezocht. In de literatuur over efficient zoeken d.m.v. SONS, komen ruwweg twee verschillende SON varianten voor:

1. *Gedeelde afstandsmaat en datastructuur.* In deze aanpak worden zoekopdrachten en expertise omschrijvingen beschreven in termen die voorkomen in een gedeelde datastructuur waarin de semantische afstanden tussen termen worden gegeven of kunnen worden afgeleid. Met semantische afstand bedoelen we de gerelateerheid van de termen m.b.t. expertise. Zo is bijvoorbeeld een expert op het gebied van tuinieren ook vaak een expert op het gebied van snoeien, waardoor de semantische afstand tussen tuinieren en snoeien klein is. Dus via deze manier, kunnen, de semantisch meest relevante computers worden gevonden voor een zoekopdracht door de termen van de zoekopdracht te vergelijken met de termen van de expertise beschrijvingen.
2. *Clustering door term overlap.* In deze aanpak beschrijven computers ook hun expertise in een aantal termen, maar zonder dat ze een gedeelde datastructuur hebben waarin ze moeten voorkomen. In deze systemen benaderen computers, vaak willekeurig, andere computers en onthouden díe systemen waarvan de expertise beschrijving het meest lijkt op de eigen beschrijvingen. Op deze manier ontstaan er dus clusters van computers die min of meer dezelfde expertise hebben. De efficientie van het zoeken wordt vervolgens bereikt door de aanname dat expertise en interesse van een gebruiker vaak gelijk zijn, en dat daarom zoekopdrachten van een gebruiker vaak kunnen worden beantwoord door díe computers die samen met de computer van de gebruiker in dezelfde cluster zitten.

Nu volgt voor ieder artikel een korte samenvatting waarbij de eerste twee artikelen een SON aanpak voorstellen gebaseerd op de eerste SON aanpak en het laatste een combinatie van de tweede SON aanpak en Distributed Hash Tables (DHTs), welke een andere manier om efficient gedecentraliseerd te zoeken.

5.2.6 Expertise gebaseerde selectie m.b.v. gedeelde ontologieën

In dit artikel wordt allereerst het *expertise-based peer selection* model geïntroduceerd welke hiervoor al genoemd is. In dit P2P sys-

teem, welke geïmplementeerd is als het routing mechanisme in Bibster [Haase et al., 2004a], worden advertentie berichten en zoekopdrachten uitgedrukt in termen die voorkomen in een, door alle computers gedeelde, ontologie. Een ontologie is een formeel omschreven, expliciete specificatie van een gedeelde conceptualisatie [Gruber, 1993]. In dit artikel wordt een bibliografisch scenario beschreven als instantie van het model waar de ontologie een verzameling is van generieke en specifieke onderwerpen binnen de informatica [acm,] gekoppeld door `subtopic` en `seeAlso` relaties. Zo is bijvoorbeeld *Object Oriented Programming* een `subtopic` van *Programming Techniques*. De computers binnen het netwerk beschrijven dus hun 'expertise' in termen die voorkomen in de gedeelde ontologie. [Li et al., 2003] beschrijft voor semantische netwerken (waarvan een ontologie een voorbeeld is), verschillende afstandsmaten om de 'semantische gerelateerdheid' tussen twee willekeurige onderwerpen te bepalen. Voor onze simulatie experimenten selecteren we de maat die het beste uit hun vergelijking komt, namelijk die maat welke het meest overeenkomt met een door mensen gegeven set van semantische afstanden (gebaseerd op intuïtie). In deze afstandsmaat wordt gekozen voor het aantal stappen die nodig zijn om binnen de semantische graaf van het ene naar het andere onderwerp te lopen. Zo is bijvoorbeeld de semantische afstand tussen *Object Oriented Programming* en *Visual Programming* kleiner dan tussen *Object Oriented Programming* en *Robotics* omdat de eerste twee bereikt kunnen worden via de directe `subtopic` relation met *Programming Techniques* en de andere twee onderling een veel langer pad hebben tot elkaar. Door deze (gedeelde) afstandsmaat zijn computers binnen het netwerk in staat om voor een zoekopdracht de semantisch dichtsbijzijnde advertenties te vinden en zo de zoekopdracht door te sturen naar de meest relevante experts. In het artikel wordt het model getest in een bibliografisch scenario, waar o.a. journals zijn gepresenteerd door computers binnen het netwerk. De dataset bestaat uit duizenden artikelen die kunnen worden gerangschikt naar onderwerp (geselecteerd uit de gedeelde ontologie) of naar conferentie/journal.

Simulatie experimenten en een veld-experiment zijn uitgevoerd om de kwaliteit van de aanpak te verifiëren. De resultaten laten zien dat de voorgestelde aanpak het aantal berichten welke nodig zijn om een bepaald percentage van de relevante computers binnen het netwerk te vinden, met minimaal een factor tien kan worden gereduceerd (t.o.v. een model dat willekeurig computers selecteert, zoals bij het Gnutella systeem).

5.2.7 Expertise gebaseerde selectie m.b.v. gedeelde similariteits matrices

In dit artikel wordt het expertise gebaseerde model geïnstantieerd op dezelfde manier als in het vorige artikel, maar dan met het verschil dat er meerdere adverteer- en zoek strategieën worden beschreven en dat de

gedeelde datastructuur geen ontologie maar een automatisch gegenereerde afstands-matrix is. Het voordeel is hierbij dus dat het maken van de gedeelde datastructuur automatisch gaat i.t.t. de vorige aanpak en daardoor minder energie vereist van de gebruikers. In deze afstands matrix staan de semantische aftanden tussen een groot aantal termen. Deze termen zijn automatisch geselecteerd door een NLP (natural language processing) programma die uit een gegeven aantal documenten de belangrijkste woorden extraheert (termen die het meest beschrijvend zijn voor de desbetreffende documenten wat betekent dat de stopwoorden en niet vaak voorkomende woorden weggelaten zijn). Op deze manier is dus ieder geselecteerde document beschreven door een aantal automatisch geselecteerde woorden. De afstands matrix wordt gegenereerd door een techniek dat 'Latent Semantic Indexing' heet [Deerwester et al., 1990] waarbij een document $\times$ term matrix wordt getransformeerd naar een term $\times$ term matrix waarbij via een statische analyse de mate van correlatie tussen de termen wordt gevonden. De afstandsmatrix wordt verdeeld onder de leden van het netwerk en er wordt vanuitgegaan dat de termen uit de matrix voldoende zijn om de expertise van de leden in uit te drukken. Daardoor is het de selectie van de documenten waaruit de matrix gegegereerd wordt belangrijk om een representatieve set te selecteren. Het doel van de simulatie experimenten is om te kijken welke combinatie van adverteer- en zoek strategiën en welke invloed de verschillende parameters hebben op de hoeveelheid berichten en aantal gevonden relevante computers in het netwerk. De resultaten laten zien dat prestaties van het algoritme niet onderdoen aan die van het algorithme dat gebruik maakt van een gedeelde ontologie.

5.2.8 Expertise gebaseerde selectie door combinatie van Distributed Hash Tables en Semantic Overlay Networks

In de vorige twee artikelen hebben we twee voorbeelden gezien van Semantic Overlay Networks die gebruik maken van een gedeelde data-structuur om de zoek opdrachten en expertise beschrijvingen in op te schrijven. Een probleem deze aanpak is dat alle computers binnen het netwerk de gedeelde datastructuur moeten downloaden of aan elkaar uitwisselen. Dit levert veel verkeer op wanneer deze datastructuur vaak veranderd zoals in het geval wanneer er frequent nieuwe onderwerpen bijkomen.

Zoals al eerder genoemd, is een alternatieve SON aanpak het z.g.n. clusteren van computers die onderling gerelateerde expertise-beschrijvingen hebben. In dit artikel wordt een intantiatie van het expertise gebaseerde model voorgesteld waar geen gebruik meer wordt gemaakt van een gedeelde data-structuur waarin de leden binnen het netwerk hun expertise beschrijven. Soms kunnen of willen computers geen samenvatting maken van hun eigen systeem om deze te delen. Daardoor kunnen ze ook niet geclusterd worden en zal daardoor de zoekopdracht willekeurig worden verspeid over

het netwerk wat zeer inefficient is. Ook voor dit probleem gaat deze aanpak een oplossing bieden.

In het artikel wat we hier bespreken, combineren we de zojuist genoemde cluster methode met een ander al bestaande methode, welke gebruik maakt van Distributed Hash Tables (DHTs). De DHT methode heeft zeer goede eigenschappen, maar ook een paar nadelen. Het doel van dit artikel is om door de clustering en DHT methode te combineren, de voordelen van beide te behouden en de nadelen die voor beide gelden op te heffen. Voordat deze methode wordt uitgelegd, volgt er eerst nog waar meer informatie over de DHT methode.

DHTs worden op dit moment gezien als belangrijke bouwstenen voor P2P systemen m.b.t. het opslaan en vinden van informatie op een volledig gedecentraliseerde manier [Aberer et al., 2003, Ratnasamy et al., 2001, Stoica et al., 2001, Rowstron and Druschel, 2001]. Het generieke idee van DHTs is dat ieder item, welke wordt gedeeld in het netwerk, automatisch een unieke identifier krijgt via een zgn. 'hash-function'. Vervolgens wordt het item (of een pointer daarnaar) efficient verstuurd naar een computer in het netwerk waarvan het adres het dichtst bij de (via een meestal nummerieke afstandsmaat) identifier van het item ligt. Met efficient wordt in dit geval bedoeld dat slechts $\emptyset(\log(N))$ berichten nodig zijn om een item op te slaan bij de juiste computer, waarbij N het aantal computers in het netwerk is. Uiteindelijk betekent dit dus dat de meeste computers in het netwerk een rijtje van identifiers van items hebben die het dichtst bij hun eigen adres liggen, inclusief de items zelf of pointers ernaar. Vervolgens kunnen die items ook weer zeer efficient teruggevonden worden: het enige wat men moet weten is de identifier van het item, welke door het DHT protocol wordt gebruikt om de computer te vinden die het item zelf (of een pointer ernaar) heeft. Op deze manier ontstaat er dus een soort gedistribueerde 'gouden-gids' functionaliteit waar eenvoudig en efficient voor een bepaald onderwerp de geregistreerde computers kunnen worden teruggevonden.

De DHT aanpak heeft, zoals gezegd, ook een aantal nadelen. Ten eerste zijn de administratie kosten om het DHT netwerk te onderhouden vrij hoog, zoals bijvoorbeeld als een computer zichzelf afmeldt van het netwerk, dan moeten andere computers de tabellen overnemen wat tot netwerk verkeer leidt. Ten tweede, meestal zijn informatie en zoekopdrachten scheef verdeeld, namelijk is het vaak zo dat bepaalde informatie veel vaker voorkomt bij computers en ook dat bepaalde zoektermen veel populairder zijn. Het kan dan voorkomen dat een bepaald systeem dat verantwoordelijk is voor bijvoorbeeld de term "Brittney Spears" veel meer verkeer te verwerken krijgt dan een systeem dat verantwoordelijk is voor de term "Semantic Web". Ten derde is de DHT aanpak erg gevoelig voor systemen die niet willen of kunnen reageren maar wel aangemeld zijn op het netwerk. Op deze manier kan het voorkomen dat de computers die zich hebben geregistreerd voor een bepaald item niet kunnen worden gevonden omdat de computer die

verantwoordelijk is voor het betreffende item, niet reageert.

In het hier besproken artikel wordt een combinatie van DHT en het clustering gebaseerde SON model voorgesteld met als doel de unieke nadelen van de individuele methodes te reduceren en de voordelen te behouden. In het algoritme wat wordt voorgesteld, worden een aantal termen uit de expertise beschrijvingen gebruikt om via DHT (dus efficient) de computers te registeren als experts op die termen. Dit betekent dat iedere computer mogelijk het register van een bepaalde term is, en daardoor verantwoordelijk voor het bijhouden van de lijst van computers, inclusief hun expertise beschrijvingen, die zich registreren voor de desbetreffende termen. Vervolgens worden deze registers geraadpleegd in het clustering proces en zoekproces. Namelijk, wanneer een computer zijn expertise kenbaar wil maken aan andere computers en ook zelf gerelateerde computers wil leren kennen, gaat het eerst naar een aantal registers die verantwoordelijk zijn voor termen die ook in zijn eigen expertise beschrijving voorkomt. Deze registers geven dan een lijst met computers terug die zichzelf hebben geregistreerd voor de desbetreffende termen. Vervolgens selecteert de computer een aantal uit deze lijst en stuurt ze een 'advertentie' bericht waarin zijn eigen expertise beschrijving staat. De computers slaan meestal de advertentie op (meestal wanneer de adverteerder een expertise beschrijving heeft die erg op de eigen beschrijving lijkt en daardoor al bij het clustering proces behoort). Naast het opslaan van de advertentie sturen ze, wanneer mogelijk, ook een lijst met gerelateerde computers terug (de verstuurder bepaalt zelf de relevantie door de expertise beschrijving van de adverteerder te vergelijken met de computers die het kent). De adverteerder slaat deze ontvangen lijst op en selecteert vervolgens weer een aantal computers uit die lijst om advertentie berichten naar toe te sturen. Dit gaat net zolang door totdat een van tevoren aangegeven aantal advertentie berichten is verstuurd. Het zoeken binnen het netwerk begint ook eerst met bij de registers als de vrager zelf nog niet geclusterd is binnen het netwerk. Dit gebeurt door het raadplegen van registers die verantwoordelijk zijn voor termen van de zoekopdracht. Deze registers reageren dan met een lijst van potentiele kandidaten die dan vervolgens door de computer die de zoekopdracht verstuurde, kunnen worden geraadpleegd. Deze kandidaten geven zelf, na raadpleging, ook zelf weer een lijst met potentiele kandidaten, zodat de zoekopdracht binnen het cluster verder kan worden doorgestuurd. Wanneer een computer al geclusterd is binnen het netwerk, hoeven de registers niet altijd meer te worden geraadpleegd voor het retourneren van een lijst van potentiele kandidaten die data hebben dat aan de zoekopdracht voldoet. Dit is meestal het geval als de zoekopdracht gerelateerd is aan de expertise van de initiator zelf. Door het clusteren, zijn namelijk de omliggende computers prima kandidaten om de vraag aan te stellen. Met andere woorden, door het clusteren hoeft er minder aanspraak te worden gemaakt op de registers, waardoor de problemen die specifiek voor de DHT aanpak verminderen. Als een computer zelf geen expertise heeft, en daar-

door niet geclusterd is, kan nog steeds efficient gezocht worden omdat de registers meestal een lijst met postentiele kandidaten hebben.

Om de hiervoor genoemde beweringen te verifiëren, zijn er simulatie experimenten uitgevoerd. De resultaten laten zien dat we minimaal dezelfde zoek efficientie als de DHT aanpak behalen, d.w.z. het aantal berichten dat nodig is om een bepaald percentage van de antwoorden binnen het netwerk te vinden, maar slechts een fractie van de DHT registers gebruiken.

De komende jaren moet blijken of de beschreven systemen in dit proefschrift werkelijk hun weg vinden via applicaties. Wat in ieder geval kan worden gezegd is dat Bibster inmiddels al behoorlijk wat positieve aandacht heeft gekregen, waarvan de Duitse 'Do-IT' software innovatie prijs ter waarde van EUR 15.000 een recent voorbeeld is.

SIKS Dissertation Series

1998

1998-1 Johan van den Akker (CWI)
DEGAS - An Active, Temporal Database of Autonomous Objects

1998-2 Floris Wiesman (UM)
Information Retrieval by Graphically Browsing Meta-Information

1998-3 Ans Steuten (TUD)
A Contribution to the Linguistic Analysis of Business Conversations within the Language/Action Perspective

1998-4 Dennis Breuker (UM)
Memory versus Search in Games

1998-5 E.W.Oskamp (RUL)
Computerondersteuning bij Straftoemeting

1999

1999-1 Mark Sloof (VU)
Physiology of Quality Change Modelling; Automated modelling of Quality Change of Agricultural Products

1999-2 Rob Potharst (EUR)
Classification using decision trees and neural nets

1999-3 Don Beal (UM)
The Nature of Minimax Search

1999-4 Jacques Penders (UM)
The practical Art of Moving Physical Objects

1999-5 Aldo de Moor (KUB) *Empowering Communities: A Method for the Legitimate User-Driven Specification of Network Information Systems*

1999-6 Niek J.E. Wijngaards (VU)
Re-design of compositional systems

1999-7 David Spelt (UT)
Verification support for object database design

1999-8 Jacques H.J. Lenting (UM)
Informed Gambling: Conception and Analysis of a Multi-Agent Mechanism for Discrete Reallocation

2000

2000-1 Frank Niessink (VU)
Perspectives on Improving Software Maintenance

2000-2 Koen Holtman (TUE) *Prototyping of CMS Storage Management*

2000-3 Carolien M.T. Metselaar (UvA)
Sociaal-organisatorische gevolgen van kennistechnologie; een procesbenadering en actorperspectie

2000-4 Geert de Haan (VU)
ETAG, A Formal Model of Competence Knowledge for User Interface Design

2000-5 Ruud van der Pol (UM)
Knowledge-based Query Formulation in Information Retrieval

2000-6 Rogier van Eijk (UU)
Programming Languages for Agent Communication

2000-7 Niels Peek (UU)
Decision-theoretic Planning of Clinical Patient Management

2000-8 Veerle Coupé (EUR)
Sensitivity Analyis of Decision-Theoretic Networks

2000-9 Florian Waas (CWI)
Principles of Probabilistic Query Optimization

2000-10 Niels Nes (CWI)
Image Database Management System Design Considerations, Algorithms and Architecture

2000-11 Jonas Karlsson (CWI)
Scalable Distributed Data Structures for Database Management

2001

2001-1 Silja Renooij (UU)
Qualitative Approaches to Quantifying Probabilistic Networks

2001-2 Koen Hindriks (UU)
Agent Programming Languages: Programming with Mental Models

2001-3 Maarten van Someren (UvA)
Learning as problem solving

2001-4 Evgueni Smirnov (UM)
Conjunctive and Disjunctive Version Spaces with Instance-Based Boundary Sets

2001-5 Jacco van Ossenbruggen (VU)
Processing Structured Hypermedia: A Matter of Style

2001-6 Martijn van Welie (VU)
Task-based User Interface Design

2001-7 Bastiaan Schonhage (VU)
Diva: Architectural Perspectives on Information Visualization

2001-8 Pascal van Eck (VU)
A Compositional Semantic Structure for Multi-Agent Systems Dynamics

2001-9 Pieter Jan 't Hoen (RUL)
Towards Distributed Development of Large Object-Oriented Models, Views of Packages as Classes

2001-10 Maarten Sierhuis (UvA)
Modeling and Simulating Work Practice BRAHMS: a multiagent modeling and simulation language for work practice analysis and design

2001-11 Tom M. van Engers (VU)
Knowledge Management: The Role of Mental Models in Business Systems Design

2002

2002-01 Nico Lassing (VU)
Architecture-Level Modifiability Analysis

2002-02 Roelof van Zwol (UT)
Modelling and searching web-based document collections

2002-03 Henk Ernst Blok (UT)
Database Optimization Aspects for Information Retrieval

2002-04 Juan Roberto Castelo Valdueza (UU)
The Discrete Acyclic Digraph Markov Model in Data Mining

2002-05 Radu Serban (VU)
The Private Cyberspace Modeling Electronic Environments inhabited by Privacy-concerned Agents

2002-06 Laurens Mommers (UL)
Applied legal epistemology; Building a knowledge-based ontology of the legal domain

2002-07 Peter Boncz (CWI)
Monet: A Next-Generation DBMS Kernel For Query-Intensive Applications

2002-08 Jaap Gordijn (VU)
Value Based Requirements Engineering: Exploring Innovative E-Commerce Ideas

2002-09 Willem-Jan van den Heuvel (KUB)
Integrating Modern Business Applications with Objectified Legacy Systems

2002-10 Brian Sheppard (UM)
Towards Perfect Play of Scrabble

2002-11 Wouter C.A. Wijngaards (VU)
Agent Based Modelling of Dynamics: Biological and Organisational Applications

2002-12 Albrecht Schmidt (UvA)
Processing XML in Database Systems

2002-13 Hongjing Wu (TUE)
A Reference Architecture for Adaptive Hypermedia Applications

2002-14 Wieke de Vries (UU)
Agent Interaction: Abstract Approaches to Modelling, Programming and Verifying Multi-Agent Systems

2002-15 Rik Eshuis (UT)
Semantics and Verification of UML Activity Diagrams for Workflow Modelling

2002-16 Pieter van Langen (VU)
The Anatomy of Design: Foundations, Models and Applications

2002-17 Stefan Manegold (UvA)
Understanding, Modeling, and Improving Main-Memory Database Performance

2003

2003-01 Heiner Stuckenschmidt (VU)
Ontology-Based Information Sharing in Weakly Structured Environments

2003-02 Jan Broersen (VU)
Modal Action Logics for Reasoning About Reactive Systems

2003-03 Martijn Schuemie (TUD)
Human-Computer Interaction and Presence in Virtual Reality Exposure Therapy

2003-04 Milan Petkovic (UT)
Content-Based Video Retrieval Supported by Database Technology

2003-05 Jos Lehmann (UvA)
Causation in Artificial Intelligence and Law - A modelling approach

2003-06 Boris van Schooten (UT)
Development and specification of virtual environments

2003-07 Machiel Jansen (UvA)
Formal Explorations of Knowledge Intensive Tasks

2003-08 Yongping Ran (UM)
Repair Based Scheduling

2003-09 Rens Kortmann (UM)
The resolution of visually guided behaviour

2003-10 Andreas Lincke (UvT)
Electronic Business Negotiation: Some experimental studies on the interaction between medium, innovation context and culture

2003-11 Simon Keizer (UT)
Reasoning under Uncertainty in Natural Language Dialogue using Bayesian Networks

2003-12 Roeland Ordelman (UT)
Dutch speech recognition in multimedia information retrieval

2003-13 Jeroen Donkers (UM)
Nosce Hostem - Searching with Opponent Models

2003-14 Stijn Hoppenbrouwers (KUN)
Freezing Language: Conceptualisation Processes across ICT-Supported Organisations

2003-15 Mathijs de Weerdt (TUD)
Plan Merging in Multi-Agent Systems

2003-16 Menzo Windhouwer (CWI)
Feature Grammar Systems - Incremental Maintenance of Indexes to Digital Media Warehouses

2003-17 David Jansen (UT)
Extensions of Statecharts with Probability, Time, and Stochastic Timing

2003-18 Levente Kocsis (UM)
Learning Search Decisions

2004

2004-01 Virginia Dignum (UU)
A Model for Organizational Interaction: Based on Agents, Founded in Logic

2004-02 Lai Xu (UvT)
Monitoring Multi-party Contracts for E-business

2004-03 Perry Groot (VU)
A Theoretical and Empirical Analysis of Approximation in Symbolic Problem Solving

2004-04 Chris van Aart (UvA)
Organizational Principles for Multi-Agent Architectures

2004-05 Viara Popova (EUR)
Knowledge discovery and monotonicity

2004-06 Bart-Jan Hommes (TUD)
The Evaluation of Business Process Modeling Techniques

2004-07 Elise Boltjes (UM)
Voorbeeldig onderwijs; voorbeeldgestuurd onderwijs, een opstap naar abstract denken, vooral voor meisjes

2004-08 Joop Verbeek (UM)
Politie en de Nieuwe Internationale Informatiemarkt, Grensregionale politiële gegevensuitwisseling en digitale expertise

2004-09 Martin Caminada (VU)
For the Sake of the Argument; explorations into argument-based reasoning

2004-10 Suzanne Kabel (UvA)
Knowledge-rich indexing of learning-objects

2004-11 Michel Klein (VU)
Change Management for Distributed Ontologies

2004-12 The Duy Bui (UT)
Creating emotions and facial expressions for embodied agents

2004-13 Wojciech Jamroga (UT)
Using Multiple Models of Reality: On Agents who Know how to Play

2004-14 Paul Harrenstein (UU)
Logic in Conflict. Logical Explorations in Strategic Equilibrium

2004-15 Arno Knobbe (UU)
Multi-Relational Data Mining

2004-16 Federico Divina (VU)
Hybrid Genetic Relational Search for Inductive Learning

2004-17 Mark Winands (UM)
Informed Search in Complex Games

2004-18 Vania Bessa Machado (UvA)
Supporting the Construction of Qualitative Knowledge Models

2004-19 Thijs Westerveld (UT)
Using generative probabilistic models for multimedia retrieval

2004-20 Madelon Evers (Nyenrode)
Learning from Design: facilitating multidisciplinary design teams

2005

2005-01 Floor Verdenius (UVA)
Methodological Aspects of Designing Induction-Based Applications

2005-02 Erik van der Werf (UM)
AI techniques for the game of Go

2005-03 Franc Grootjen (RUN)
A Pragmatic Approach to the Conceptualisation of Language

2005-04 Nirvana Meratnia (UT)
Towards Database Support for Moving Object data

2005-05 Gabriel Infante-Lopez (UVA)
Two-Level Probabilistic Grammars for Natural Language Parsing

2005-06 Pieter Spronck (UM)
Adaptive Game AI

2005-07 Flavius Frasincar (TUE)
Hypermedia Presentation Generation for Semantic Web Information Systems

2005-08 Richard Vdovjak (TUE)
A Model-driven Approach for Building Distributed Ontology-based Web Applications

2005-09 Jeen Broekstra (VU)
Storage, Querying and Inferencing for Semantic Web Languages

2005-10 Anders Bouwer (UVA)
Explaining Behaviour: Using Qualitative Simulation in Interactive Learning Environments

2005-11 Elth Ogston (VU)
Agent Based Matchmaking and Clustering - A Decentralized Approach to Search

2005-12 Csaba Boer (EUR)
Distributed Simulation in Industry

2005-13 Fred Hamburg (UL)
Een Computermodel voor het Ondersteunen van Euthanasiebeslissingen

2005-14 Borys Omelayenko (VU)
Web-Service configuration on the Semantic Web; Exploring how semantics meets pragmatics

2005-15 Tibor Bosse (VU)
Analysis of the Dynamics of Cognitive Processes

2005-16 Joris Graaumans (UU)
Usability of XML Query Languages

2005-17 Boris Shishkov (TUD)
Software Specification Based on Re-usable Business Components

2005-18 Danielle Sent (UU)
Test-selection strategies for probabilistic networks

2005-19 Michel van Dartel (UM)
Situated Representation

2005-20 Cristina Coteanu (UL)
Cyber Consumer Law, State of the Art and Perspectives

2005-21 Wijnand Derks (UT)
Improving Concurrency and Recovery in Database Systems by Exploiting Application Semantics

2006

2006-01 Samuil Angelov (TUE)
Foundations of B2B Electronic Contracting

2006-02 Cristina Chisalita (VU)
Contextual issues in the design and use of information technology in organizations

2006-03 Noor Christoph (UVA)
The role of metacognitive skills in learning to solve problems

2006-04 Marta Sabou (VU)
Building Web Service Ontologies

2006-05 Cees Pierik (UU)
Validation Techniques for Object-Oriented Proof Outlines

2006-06 Ziv Baida (VU)
Software-aided Service Bundling - Intelligent Methods & Tools for Graphical Service Modeling

2006-07 Marko Smiljanic (UT)
XML schema matching – balancing efficiency and effectiveness by means of clustering

2006-08 Eelco Herder (UT)
Forward, Back and Home Again - Analyzing User Behavior on the Web

2006-09 Mohamed Wahdan (UM)
Automatic Formulation of the Auditor's Opinion